MirrorFlow:
Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation
– Supplementary Material –

Junhwa Hur Stefan Roth
Department of Computer Science, TU Darmstadt

We here provide additional details on the data term, an analysis of the optimizer, an accuracy analysis in occluded regions, and details on the processing time.

A. Details on the Data Term

In Eqs. (2b) and (2c) of the main paper, the function $\rho_{\mathrm{D}}(\mathbf{p}, \mathbf{H}_{s_\mathbf{p}})$ measures the photometric error between a pixel $\mathbf{p}$ and its corresponding pixel $\mathbf{H}_{s_\mathbf{p}}\mathbf{p}$ in the other frame. For example, given a homography motion $\mathbf{H}^f_{s_\mathbf{p}}$, the data cost for pixel $\mathbf{p}$ in I^t is given as

$$\rho^f_{\mathrm{D}}(\mathbf{p}, \mathbf{H}^f_{s_\mathbf{p}}) = \min\left\{\rho_l\big(\phi(\mathbf{p}, \mathbf{H}^f_{s_\mathbf{p}})\big), \tau_{\mathrm{D}}\right\} \tag{7a}$$

with

$$\phi(\mathbf{p}, \mathbf{H}^f_{s_\mathbf{p}}) = \tag{7b}$$

$$\alpha_D \sum_{\mathbf{y}\in\{-3,\dots,3\}^2} f\Bigg(\underbrace{T\big(I^t(\mathbf{p}+\mathbf{y}) - I^t(\mathbf{p})\big)}_{\text{ternary value at } \mathbf{p} \text{ in } I^t} - \underbrace{T\big(I^{t+1}(\mathbf{H}^f_{s_\mathbf{p}}(\mathbf{p}+\mathbf{y})) - I^{t+1}(\mathbf{H}^f_{s_\mathbf{p}}\mathbf{p})\big)}_{\text{ternary value at } \mathbf{H}^f_{s_\mathbf{p}}\mathbf{p} \text{ in } I^{t+1}}\Bigg)$$

$$+(1-\alpha_D)\underbrace{|\nabla I^{t+1}(\mathbf{H}^f_{s_\mathbf{p}}\mathbf{p}) - \nabla I^t(\mathbf{p})|}_{\text{gradient constancy penalty}}, \tag{7c}$$

which is the weighted sum of the ternary transform and gradient constancy penalty.

Deviations are penalized by a Lorentzian penalty $\rho_l(x) = \alpha_l \log((1+x^2)/2\sigma_l^2)$, truncated at τ_{D}. The idea behind function $\phi(\mathbf{p}, \mathbf{H}^f_{s_\mathbf{p}})$ is to calculate the Hamming distance of two 7×7 ternary patches, one around pixel $\mathbf{p}$ in I^t and one around the corresponding pixel $\mathbf{H}^f_{s_\mathbf{p}}\mathbf{p}$ in I^{t+1}. Unlike the conventional ternary transform [34], we use a continuous variant. Specifically, we relax the definition of the Hamming distance and adopt the sigmoid function

$$T(x) = \frac{2}{1+\exp(-\sigma_T x)} - 1 = \frac{1-\exp(-\sigma_T x)}{1+\exp(-\sigma_T x)} \tag{8}$$

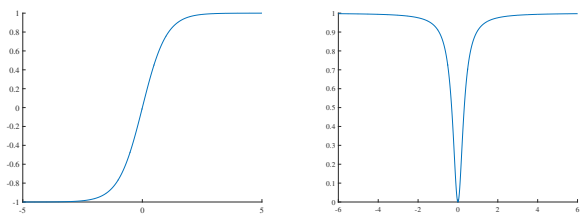
instead of a true ternary value, and use the Geman-McClure function [54] to score the differences in the ternary signature between the patches:

$$f(x) = \frac{x^2}{(\sigma_f + x^2)}. \tag{9}$$

As shown in Figs. 6a and 6b, these continuous functions approximate the conventional discrete setting, but they assess subtle brightness variations more naturally when their input is near zero. In other words, they are still robust, but less brittle than the original Hamming-based definition.

Furthermore, when calculating the ternary value at point $\mathbf{H}^f_{s_\mathbf{p}}\mathbf{p}$ in the other frame, we calculate it not on the conventional ternary patch that is centered at the transformed point, but on a transformed patch. Eq. (7b) precisely expresses how to calculate the ternary value on the warped patch (*i.e.*, referring the intensity at $\mathbf{H}^f_{s_\mathbf{p}}(\mathbf{p}+\mathbf{y})$ instead of $\mathbf{H}^f_{s_\mathbf{p}}\mathbf{p}+\mathbf{y}$). Similar to a classical iterative-warping scheme, this strategy yields a more comprehensive data cost that is invariant to local shape deformation caused by the motion.

We observe that the two practices above increase the flow accuracy. Table 4 compares the flow accuracy of three different ways of calculating the ternary value: *(i)* our standard implementation including both the continuous ternary variant and the patch transformation (*standard*), *(ii)* the standard implementation with the conventional discrete setting of the ternary transform (*discrete*) but with patch transformation, *(iii)* the standard implementation without the patch



(a) (b)

Figure 6. *(a)* Sigmoid function. *(b)* Geman-McClure function.

Method	Non-occluded pixels			All pixels		
	Fl-bg	Fl-fg	Fl-all	Fl-bg	Fl-fg	Fl-all
standard	**6.52 %**	**11.72 %**	**7.41 %**	**9.26 %**	**13.94 %**	**9.98 %**
discrete	7.11 %	12.26 %	7.99 %	9.88 %	14.57 %	10.60 %
w/o transformation	7.29 %	12.35 %	8.16 %	10.41 %	14.80 %	11.08 %

Table 4. Evaluation of different methods for computing the ternary census on the KITTI training set. See text for details.

transformation (*w/o transformation*) but with the continuous variant.

As presumed, using the continuous ternary variant and the transformed ternary patch both yield better accuracy by reducing the number of flow errors by about 5.85 % and 9.93 % respectively. This experiment has been conducted on the KITTI Optical Flow 2015 training set.

B. Analysis of the Optimizer

As discussed in Sec. 3.3.1 of the main paper, we first collect a set of proposals when assigning homography motions for each superpixel, and sequentially run expansion moves on each subgraph of superpixels to allow for an efficient optimization. We assemble a set of subgraphs in a way that each subgraph consists of neighboring 30 superpixels with 70 % overlap between each other. We empirically found that this is an advantageous setting in our problem.

Choosing the number of superpixels in each subgraph affects both flow accuracy and energy at convergence. When the number of superpixels is high, proposals can be propagated into broader regions, but the algorithm has a smaller chance of finding locally optimal homography motions, which results in slower convergence. On the other hand, when the number of superpixels is low, locally optimal motions can lower the energy level more quickly, but the optimization can get stuck in local optima as it propagates labels only in small regions, which eventually leads to higher flow error rates.

Figure 7 and Fig. 8 demonstrate the energy and the flow error rate (on KITTI 2015 training), respectively, versus the processing time, depending on the number of superpixels in each subgraph. Each dot on a graph represents an iteration step. These two figures illustrate the trade-off described above. We found that having 30 superpixels for each subgraph yields the lowest flow error rates.

Choosing the size of overlapping regions between subgraphs also incurs a trade-off: When the size is getting bigger, the proposals can be propagated more effectively between subgraphs, which helps finding lower energy solutions in fewer iterations. However, it requires more processing time per iteration because the size of the subgraphs is increased. When the size of overlaps gets smaller, on the other hand, less processing time per iteration is needed, but the optimizer propagates proposals through subgraphs less effectively, leading to more iterations being required.

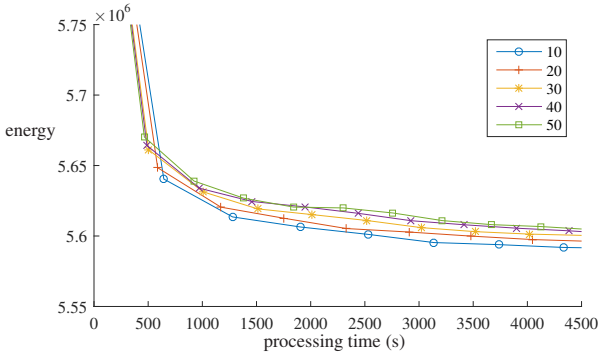


Figure 7. Overall energy depending on the number of superpixels in each subgraph.

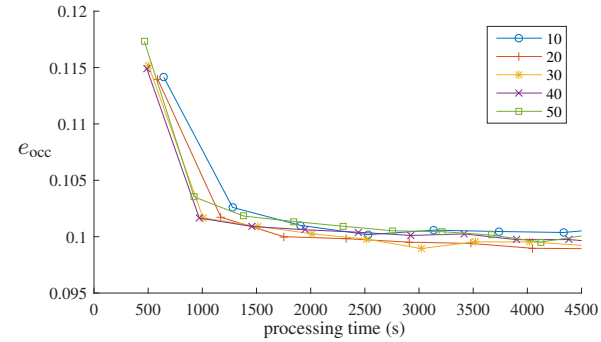


Figure 8. The estimated flow error rates depending on the number of superpixels in each subgraph.

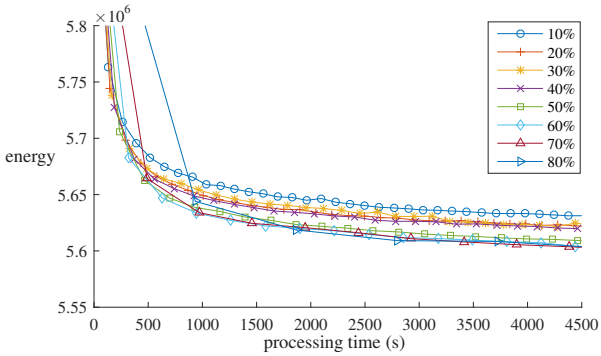


Figure 9. Overall energy depending on the overlap setting.

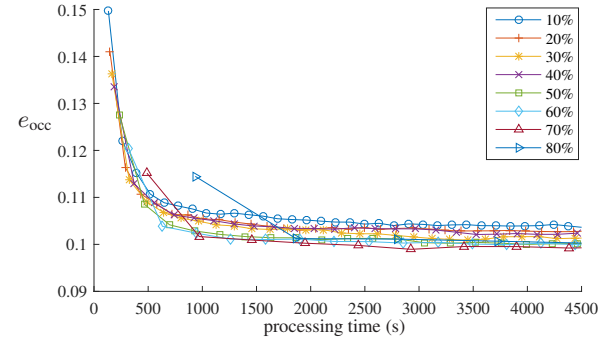


Figure 10. The estimated flow error rates depending on the overlap setting.

Figure 9 and Fig. 10 demonstrate the energy and the flow error rate (on KITTI 2015 training), respectively, versus processing time, depending on the overlap size between subgraphs. As in Fig. 9, if the overlap size is more than 50 %, the energy is converging to a lower value, but with a similar speed. Figure 10 demonstrates that having 70 % of overlap between subgraphs yields the lowest flow error rates in the same processing time. However, please note that the flow accuracy differences between the settings are not very significant (< 5 %).

C. Performance in Occluded Regions

We analyze the flow estimation accuracy of top-performing algorithms including ours especially in occluded regions on the KITTI Optical Flow 2015 benchmark [12]. Unlike the MPI Sintel Flow Dataset [8], the KITTI benchmark does not explicitly provide the statistics for occluded areas. Thus, we indirectly deduce them.

To that end, let us define the variables $n_{\text{all}}, n_{\text{noc}}, n_{\text{occ}}, e_{\text{all}}, e_{\text{noc}}$, and e_{occ} as follows:

- n_{all} : no. of all pixels considered in evaluation
- n_{noc} : no. of non-occluded pixels
- n_{occ} : no. of occluded pixels
- e_{all} : no. of all pixels with an incorrect flow estimate
- e_{noc} : no. of non-occluded pixels with an incorrect flow estimate
- e_{occ} : no. of occluded pixels with an incorrect flow estimate.

Then, the following equations naturally hold:

$$n_{\text{all}} = n_{\text{noc}} + n_{\text{occ}} \tag{10a}$$

$$e_{\text{all}} = e_{\text{noc}} + e_{\text{occ}}. \tag{10b}$$

We are interested in estimating the flow error rate in occluded areas, $e_{\text{occ}}/n_{\text{occ}}$. From Eqs. (10a) and (10b) we have

$$\frac{e_{\text{occ}}}{n_{\text{occ}}} = \frac{e_{\text{all}} - e_{\text{noc}}}{n_{\text{all}} - n_{\text{noc}}} = \frac{\frac{e_{\text{all}}}{n_{\text{all}}} - \frac{e_{\text{noc}}}{n_{\text{all}}}}{1 - \frac{n_{\text{noc}}}{n_{\text{all}}}} = \frac{\frac{e_{\text{all}}}{n_{\text{all}}} - \frac{e_{\text{noc}}}{n_{\text{noc}}}\frac{n_{\text{noc}}}{n_{\text{all}}}}{1 - \frac{n_{\text{noc}}}{n_{\text{all}}}}. \tag{11}$$

Given that we do not know the ratio of non-occluded pixels, we substitute $n_{\text{noc}}/n_{\text{all}}$ with α in Eq. (11) and obtain

$$\frac{e_{\text{occ}}}{n_{\text{occ}}} = \frac{1}{1-\alpha}\frac{e_{\text{all}}}{n_{\text{all}}} - \frac{\alpha}{1-\alpha}\frac{e_{\text{noc}}}{n_{\text{noc}}}, \tag{12}$$

where the values $e_{\text{all}}/n_{\text{all}}$ and $e_{\text{noc}}/n_{\text{noc}}$ are the flow error rates on all pixels and non-occluded pixels, respectively, which can be found in Table 1 of the main paper. Therefore, we can indirectly infer the flow error rate in occluded areas based on the statistics from Table 1 and in terms of $\alpha = n_{\text{noc}}/n_{\text{all}}$, which denotes the (unknown) ratio of the number of non-occluded pixels to that of all pixels that are considered in the evaluation.

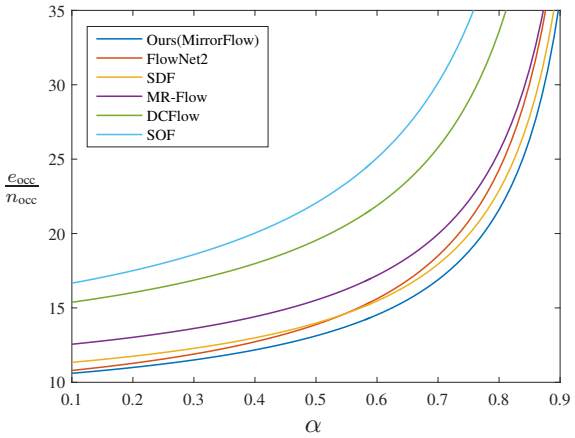


Figure 11. The estimated flow error rates of top-performing algorithms in occluded regions with respect to $\alpha = n_{\text{noc}}/n_{\text{all}}$.

Method	Fl-all in occluded pixels (estimates)
Ours (MirrorFlow)	**28.19 %**
SDF [2]	29.80 %
FlowNet2 [19]	32.36 %
MR-Flow [46]	36.23 %
DCFlow [48]	44.47 %
SOF [33]	54.33 %

Table 5. Estimated flow errors for occluded pixels (when $\alpha = 0.8635$). Our method demonstrates the lowest error among all published two-frame methods on the KITTI benchmark.

In Fig. 11 we now plot the estimated flow error rates of top-performing algorithms in occluded regions by varying the unknown ratio α. We observe that our MirrorFlow approach consistently shows the lowest error rates among the top-performing algorithms regardless of values of the ratio α. Considering that the ratio α for the KITTI 2015 training set is $\alpha = 0.8635$, we can confidently infer that our method demonstrates the lowest optical flow error among the top-performing algorithms on the KITTI benchmark. Table 5 gives the estimated results assuming the same α as on the training set.

D. Processing Time

For processing a 1226×370 image, the algorithm takes around 40 minutes on a single core until the accuracy no longer increases (tested on Intel Xeon CPU E5-2650 2.20GHz). Yet, the algorithm can be easily parallelized because the local subgraphs that do not overlap with each other can be processed at the same time [40]. Using 4 cores, the runtime decreases down to 11 minutes.

The main bottleneck is calculating the ternary transform in the data term. We calculate the ternary census on transformed patches, which needs to be done for every different homography motion. When just using a plain data term (penalizing intensity + gradient differences), the runtime is only 4 minutes. CNN-based learned descriptors also have

the potential to lead to a speedup as a future work.

References

[54] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Int. J. Comput. Vision*, 19(1):57–91, July 1996. 1