Determining Motion Boundaries From Image Boundaries Using Deep Descriptor Matching

Author: Maxim Tatarchenko
Supervisor: Eddy Ilg

28th September, 2014

Contents

1 Introduction **2**

2 Approach **3**
2.1 Outline 3
2.2 Response maps 6
2.3 Correspondences 9

3 Evaluation **11**

4 Conclusions **15**

Bibliography **16**

Chapter 1

Introduction

Image boundaries, as stated in [3], are pixels that mark the transition from one relatively constant property region to another. In the case of optical flow estimation we are particularly interested in motion boundaries. This project presents an approach to extract initial motion boundary clues from image boundaries. We then use the detected motion boundaries to modify the diffusivity term of the energy functional from [1] to account for motion discontinuities to improve optical flow estimation.

The overall idea is to perform global sparse descriptor matching along the two sides of a boundary using the deep matching approach from [4]. To obtain the initial boundaries from the images we use the generalized boundary detector from [3]. In a first step, the extracted boundaries are disconnected into lists of boundary segments. On the two sides of each boundary segment, which we arbitrarily label A and B, we first compute HOG descriptors. The general approach is then to match the A and B sides of the segments separately.

If both sides of the boundary segment from the first image match to a single boundary segment in the second image, the segment is not a motion boundary as both sides move simultaneously. This means, that the segment should be excluded from the boundary map. Otherwise, the segment can be a motion boundary and we consider it to be important for optical flow estimation. In theory, the approach further makes it possible to determine which side of the motion boundary belongs to the foreground and which to the background. However, this has some limitations in practice.

Chapter 2

Approach

2.1 Outline

In a first step, we extract boundary maps from two consecutive video frames (denoted I^t and I^{t+1} respectively) using the generalized boundary detector from [3]. To only consider meaningful boundaries, we first remove all the values from the maps that are below a certain threshold.

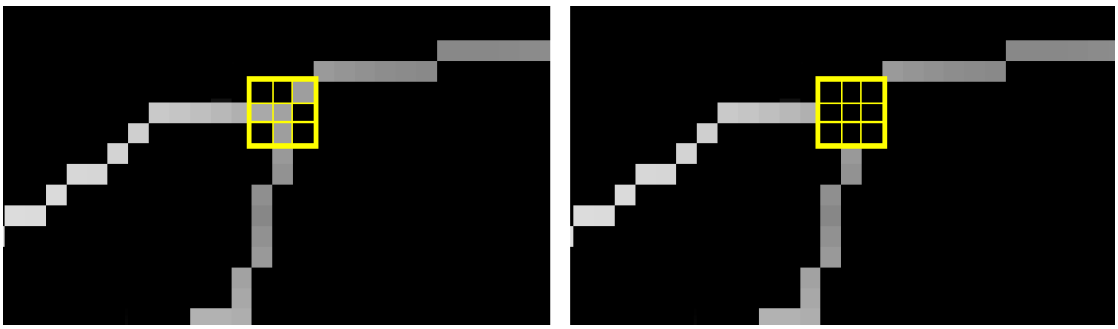

Figure 2.1: If the mask is centered on a non-zero pixel which has more than two non-zero neighbors (**left**), there is a branch and all the pixels in the mask are set to zero (**right**).

Boundaries in such maps may have branches, which makes it hard to define sides and to perform matching afterwards. Therefore, we preprocess boundary maps to obtain two lists of disconnected boundary segments. To find and remove the branches, we scan the boundary map with a 3x3 filter mask. Whenever the central element of the mask is non-zero and has more then two non-zero neighbors, there is a branch in the boundary map. This is illustrated in Figure 2.1. To remove it, we set the entire pixels within the mask to zero.

The next step is to compute descriptor values along each boundary segment on both sides (see Figure 2.2). Similar to [2], we use HOG descriptors consisting of 9 local histograms. Each gradient histogram comprises 15 different orientations and is computed in a 7x7 neighborhood. Thus, each descriptor is represented as a 135-dimensional vector. We want every single descriptor to only capture information from one side of the boundary segment (denoted A and B). To achieve this, we transform boundary segments into straight lines and warp the image around them respectively, as illustrated in Figure 2.3. Descriptor values are then computed on the warped images (see Figure 2.4).

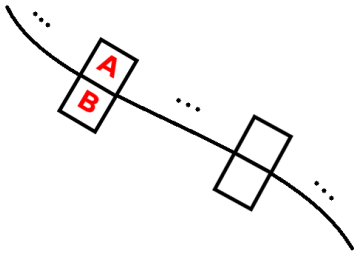

Figure 2.2: Descriptors are placed along the boundary segment on both sides.

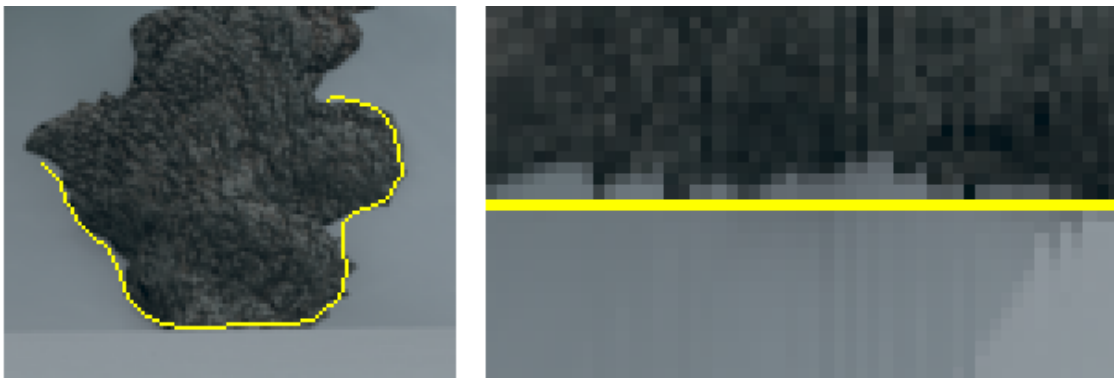

Figure 2.3: Boundary segments (yellow) from original images (**left**) are transformed into straight lines and images are warped respectively. Warped images (**right**) are then used for descriptor computation.

After that, we want to compute similarities between descriptors along both sides of the warped boundary segments. Following the deep matching approach, we then construct response maps. These are images containing similarities between a certain descriptor from I^t and all the descriptors from I^{t+1}. To capture information about different scales in the image and avoid local maxima during the matching step later, it makes sense to consider descriptors of different sizes (here denoted levels). Higher-level descriptors are computed by aggregating lower-level descriptors, as illustrated in Figure 2.5. Details of response map computation are presented in Section 2.2.

Next, we aggregate responses from different levels of response pyramids to obtain a set of matches for reference points of I^t (see Section 2.3 for a detailed description). Using voting, we then find final correspondences for each side of whole boundary segments. The voting score for a boundary segment is the sum of weights of matches coming from different pyramid levels and pointing to this segment. Based on these final correspondences, the decision is made: if both sides of the segment from I^t point to the same segment in I^{t+1}, this means the boundary segment is not a motion boundary and it can be excluded from the boundary map used for optical flow estimation. Otherwise, the segment is a potential motion boundary and is therefore preserved.

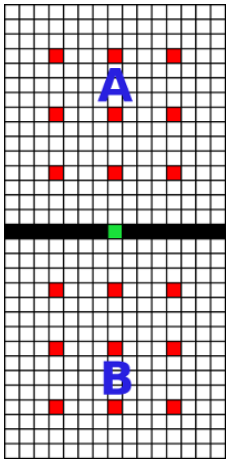

Figure 2.4: The HOG descriptors are computed on the warped images as a concatenation of 9 local orientation histograms.

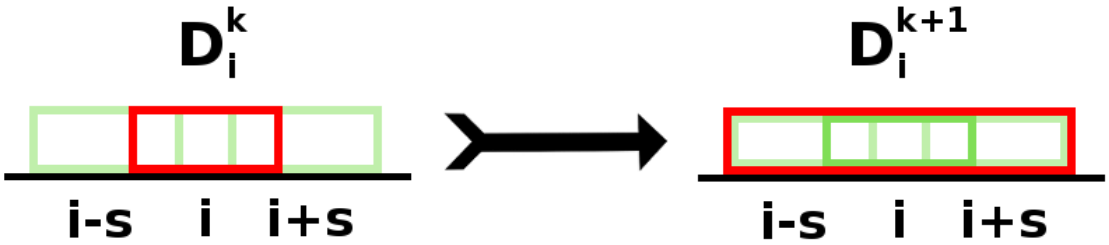

Figure 2.5: To consider different scales in the image during response map computation, we use descriptors of different sizes (marked red). A descriptor for pixel i on a higher level (D_i^{k+1}) is computed as an aggregation of lower-level responses of neighboring descriptors with indexes $\{i, i-s, i+s\}$.

2.2 Response maps

Response maps represent similarities between a certain descriptor from I^t and all the descriptors from I^{t+1}. To further capture information from different scales in the image, pyramids of response maps are computed which we name response pyramids. Let us follow the complete process of response pyramid computation and consider one boundary segment c_i^t from the set of boundary segments C^t of I^t. We want to find a corresponding boundary segment for c_i^t in the set of boundary segments C^{t+1} of I^{t+1}. For every reference pixel j of c_i^t we compute a response map $S_{i,j}^0$ of level 0 consisting of response values.

Each entry of $S_{i,j}^0$ is the similarity between the descriptor for pixel j of the boundary segment i from I^t and some other descriptor from I^{t+1}. To calculate response values, we use the non-negative cosine similarity function on descriptor vectors (denoted $Desc_t$ and $Desc_{t+1}$ for descriptors from I^t and I^{t+1} respectively).

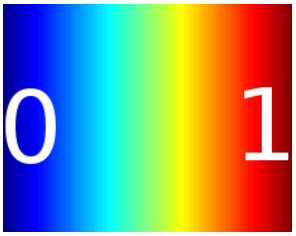

Figure 2.6: Jet color-coding is used to visualize response maps. Red corresponds to high similarities, blue to low ones.

$$sim = \frac{\langle Desc_t, Desc_{t+1} \rangle}{\|Desc_t\| \cdot \|Desc_{t+1}\|}, 0 \leq sim \leq 1 \quad (2.1)$$

The cosine similarity has the advantage of being in the range [0..1]. In general, it is possible to use some other measure to compute similarities (e.g. Euclidean distance). Instead of computing response maps densely for each pixel j of c_i^t, we only consider each s-th pixel, s denoting the stride. We visualize response values using jet color-coding (see Figure 2.6). An example of the response pyramid computation for a pixel is shown in Figure 2.7.

Following the approach from [4], we propagate initial responses from the zero level to higher levels. At each level response maps are first max-pooled to account for small deformations. For computational reasons, the response maps are then subsampled. Afterwards, adjacent responses from level $k-1$ are aggregated as follows to obtain the new response map of level k:

$$S_{i,j}^k = S_{i,j-s}^{k-1} + S_{i,j}^{k-1} + S_{i,j+s}^{k-1} \quad (2.2)$$

Similar to [4], to better propagate responses from lower to higher levels and make them more distinct, we apply a power transformation $(S_{i,j}^k)^\lambda$ with $\lambda = 1.6$. All these steps are summarized in Algorithm 2.1.

Algorithm 2.1: Computing response maps for every boundary segment of I^t

input : $C^t = \{c^t\}$, $C^{t+1} = \{c^{t+1}\}$ - Sets of boundary segments
output: $\{S^k_{i,j}\}$, $i = 1..|C^t|, j = 1..|c^t_j|$ - A set of response maps
s = 8
foreach pixel j of c^t_i in C^t stride s **do**
    Compute initial response map $S^0_{i,j}$:
    foreach pixel y of c^{t+1}_x in C^{t+1} **do**
        $S^0_{i,j}(x, y) = \text{sim}(Desc^t(i, j), Desc^{t+1}(x, y))$
    end
end
$k = 1$
Set descriptor size: $sz = 16$
Compute response maps of higher levels:
while $sz < |c^t_j|$ **do**
    foreach pixel j of c^t_i in C^t step s **do**
        Max-Pool($S^{k-1}_{i,j}$)
        Subsample($S^{k-1}_{i,j}$)
        $S^k_{i,j} = S^{k-1}_{i,j-s} + S^{k-1}_{i,j} + S^{k-1}_{i,j+s}$
        $S^k_{i,j} = (S^k_{i,j})^\lambda$
    end
    $s = s * 2$
    $k = k + 1$
    $sz = sz * 2$
end

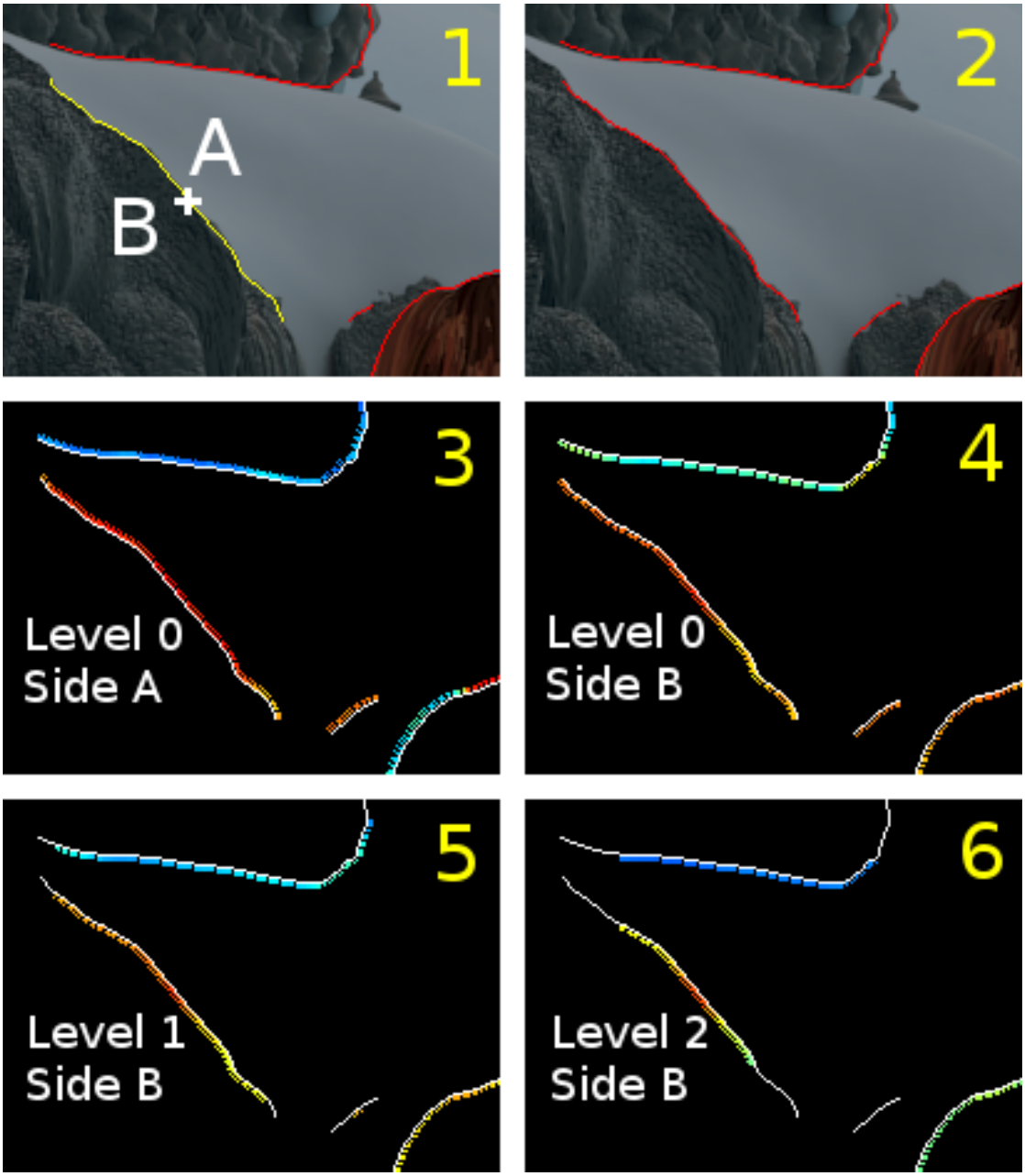

Figure 2.7: Response pyramid computation between a point descriptor (white cross) from I^t (**1**) and all the descriptors along the boundaries of I^{t+1} (**2**). High responses for side A (**3**) where descriptors capturing snow are compared. For side B responses on levels 1, 2 (**5,6**) are more distinct than on level 0 (**4**).

2.3 Correspondences

At this step of the process, we convert similarity information stored in response maps into point-to-point correspondences. For each reference point from I^t we find a set of weighted correspondence hypotheses. These hypotheses are computed hierarchically, the ones originating at higher levels of response pyramids are considered to describe larger areas and thus to be more important.

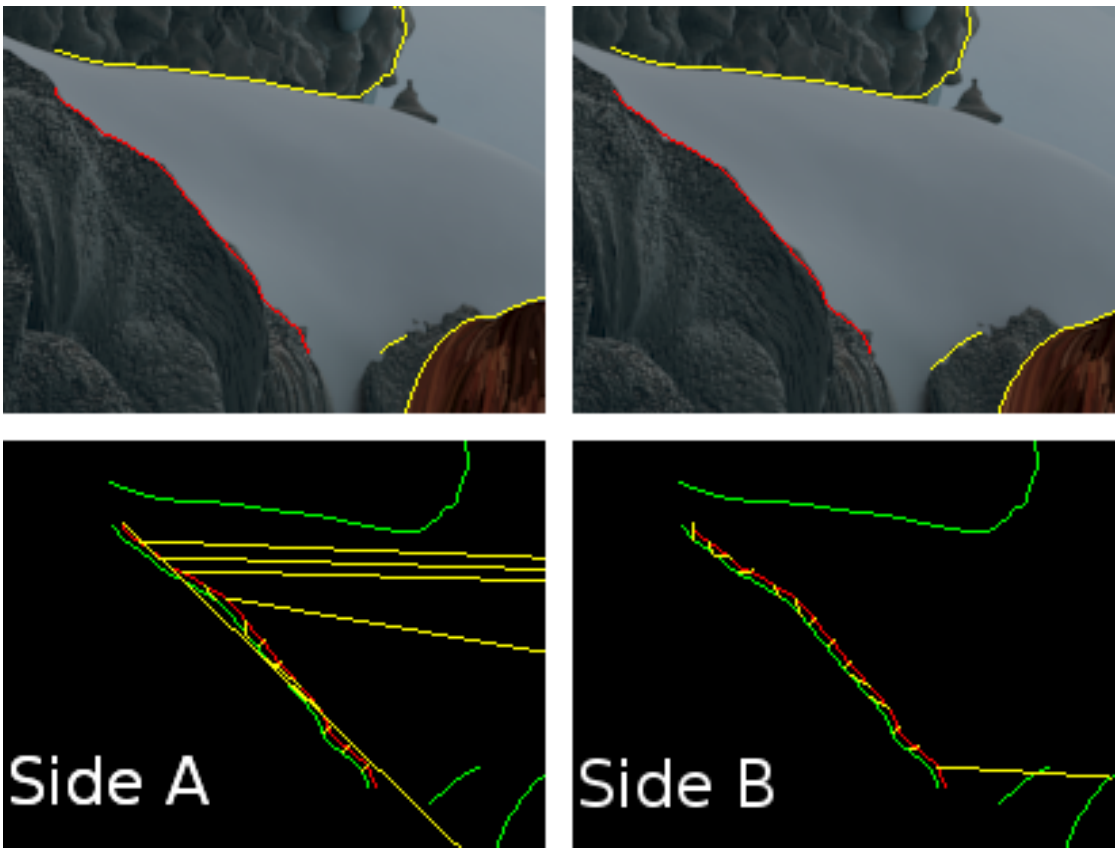

Figure 2.8: An example of final correspondences for reference points of a boundary segment. The top row shows initial images with marked boundary segments. The actually corresponding segments are shown in red. The bottom row shows the estimated correspondences for the boundary segment of interest from I^t (red in bottom images). All the boundary segments from I^{t+1} are shown in green. There are some outliers on side A due to the homogeneous background. Side B matches almost perfectly.

In each response map for each boundary segment, we find the maximal response value and propagate this information down the pyramid (backward propagation)

to get correspondences between single reference pixels belonging to the respective patch. During the backward propagation it is important to follow the correct paths that emerged during max-pooling and subsampling.

Weights for correspondences are computed according to the following heuristic:

$$w = sim \cdot (l + 1) \cdot \hat{S}, \tag{2.3}$$

where l is the level of the response pyramid at which the correspondence originates and $\hat{S}$ is the value of the maximum in this response map, while sim is the bottom level descriptor similarity from (2.1). Based on the sets of weights $\{w_k\}$, we can find one global best correspondence for each reference pixel (illustrated in Figure 2.8).

All the obtained correspondence sets are then used to find matches between whole boundary segments. Let us consider segment c_i^t and its set of correspondences pointing to different segments of C^{t+1}. Using voting, we select the best matching boundary segments for both the A and B sides of c_i^t denoted c_{jA}^{t+1} and c_{jB}^{t+1} respectively. Having $jA = jB$ means that both sides of the bondary segment moved together and we should not consider this segment as a motion boundary. If $jA \neq jB$, the segment is considered to be a potential motion boundary and is included into the output boundary map used later for optical flow estimation.

Chapter 3

Evaluation

The proposed approach was evaluated on a subset of images from the Sintel dataset containing the most complicated cases. In general, there was a significant increase in performance compared to standard LDOF with the deep matching. Evaluation results are presented in Table 3.1. The numbers show the estimated end-point-error in different cases: the original LDOF [1] implementation with matches from deep matching [4] (**DM**) extended by our motion boundary detector (**+MBD**) and occlusion check (**+OCC**) or both (**+OCC + MBD**). The occlusion check is performed by allowing the sources of a target pixel to only come frome one pixel cluster. If a target sources from different clusters, the source with the lower error is kept and the source with the higher error is marked as occluded. Occluded pixels are then disabled in the data term during energy minimization.

On average, the accuracy increases when adding motion boundary information in both cases, with or without occlusion check (improvement is 10% and 4% respectively). This shows that it is beneficial to use the occlusion check and the motion boundary detector in combination. In general, the algorithm tends to underestimate the correct solution. We use a quite simple heuristic to make decisions about a segment being a motion boundary or not (both sides correspond to the same or different boundaries). Many motion boundaries are not detected this way, because for small motions, both sides can still be matched.

Figure 3.1: Homogeneous background makes it hard to find matches robustly.

	DM	+MBD	+OCC	+OCC +MBD
Image 1	10.6535	14.2146	7.6880	7.4799
Image 2	43.1911	35.9326	45.2541	31.3443
Image 3	6.5338	6.4594	6.6542	7.1010
Image 4	3.6892	3.0682	4.5372	2.9283
Image 5	38.1519	38.9580	37.4398	37.9217
Image 6	47.3305	39.3391	50.9888	37.9868
Image 7	52.9657	52.0020	53.1377	51.4845
Image 8	16.6995	14.7963	17.2986	14.6974
Image 9	11.8467	13.5595	11.3391	9.0520
Image 10	12.2100	14.7699	12.8212	16.5183
Image 11	8.4036	8.0631	7.7931	7.3595
Average error	22.8796	21.9239	23.1774	20.3522

Table 3.1: Optical flow estimation end-point-error on images from the Sintel dataset. **DM** column contains results for standard LDOF with matches from deep matching [4]. Other columns show results with added motion boundary detector (**+MBD**), occlusion check (**+OCC**) and both (**+OCC +MBD**).

Another limitation of the approach is its inability to adequately handle homogeneous background. For example, many frames from the Sintel dataset contain snow, which has no structure and is almost the same color everywhere (see Figure 3.1). This means that every descriptor capturing information about a part of such background will have high responses in many areas of the image. Such an ambiguity can lead to mismatches. There is also a problem related to the limited accuracy of the boundary detector. The Gb algorithm finds boundary estimates, that can significantly deviate from the true boundaries (see Figure 3.2). In some cases this also leads to errors in descriptor matching.

Figure 3.2: Boundaries estimated by Gb can significantly deviate from the actual boundaries in the image.

An example result for a pair of images is illustrated in Figure 3.3. The estimated flow field becomes more accurate. Blur on the motion boundaries is partially removed. However, on some images there is a decrease in performance. An illustration of such a case is shown in Figure 3.4. Due to occlusion, it is not possible to

find the correct match for a certain boundary segment. This means the matches of A and B sides point to different boundary segments and the boundary is considered a motion boundary although it is not. This then leads to errors in optical flow estimation.

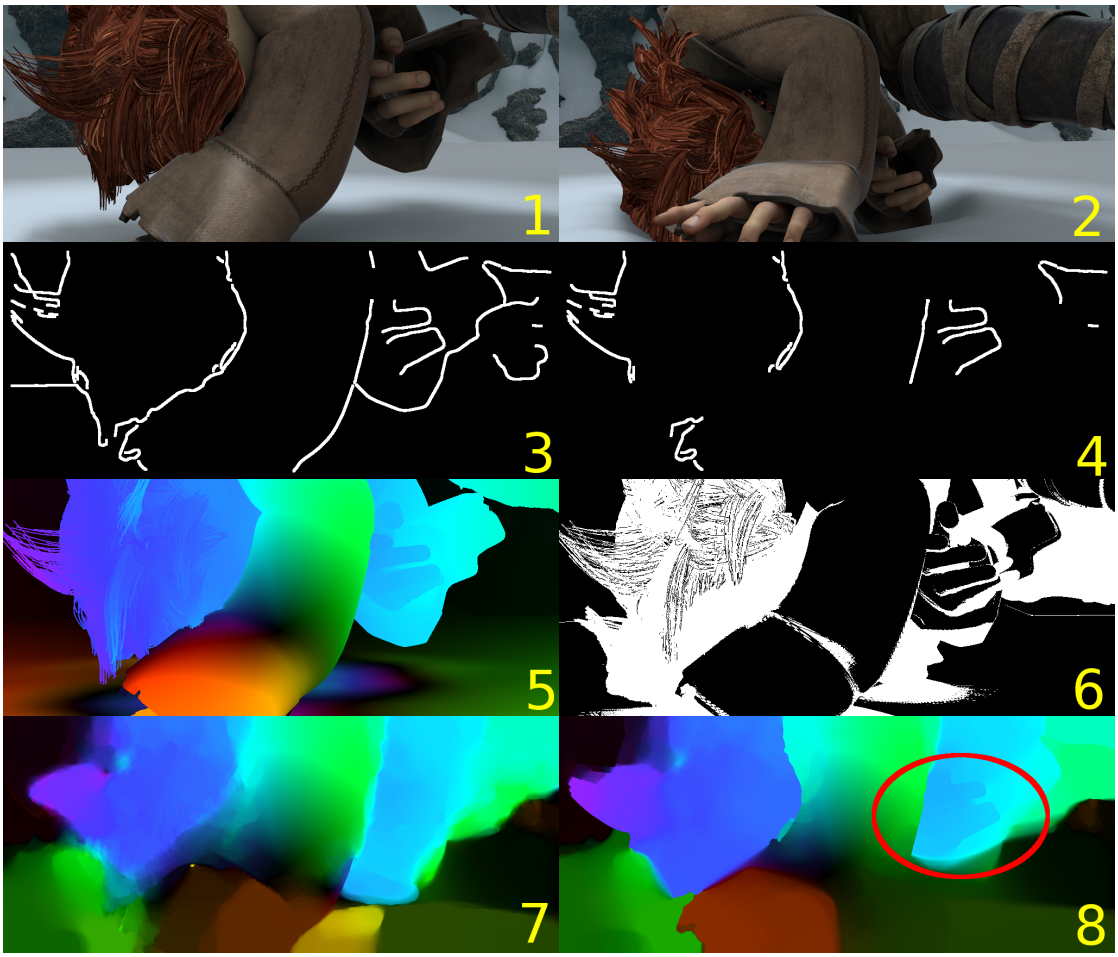

Figure 3.3: Example output for a pair of images (**1** and **2**). The initial boundary map after preprocessing is shown in **3** and the final one containing only motion boundaries in **4**. The ground truth optical flow field is shown in **5**, the occlusions estimated with occlusion check in **6**. The flow field shown in **8** was estimated using information about the motion boundaries and occlusions. It is clearly more accurate than the flow field estimated using standard LDOF with the deep matching shown in **7**. The flow on the motion boundaries becomes less blurred. Some finer details (i.e. hand marked by a red ellipse in **8**) are more clear.

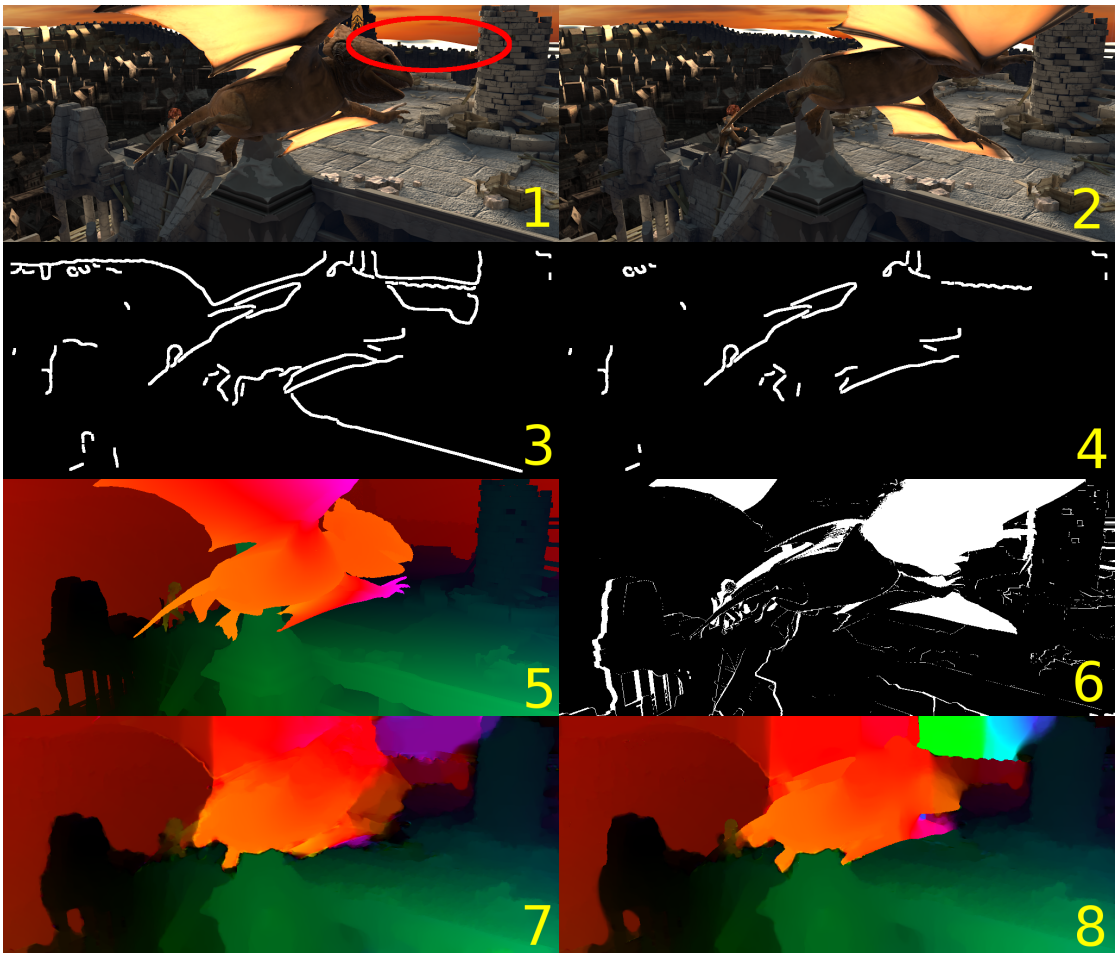

Figure 3.4: Example output for a pair of images (**1** and **2**). The initial boundary map after preprocessing is shown in **3** and the final one containing only motion boundaries in **4**. The ground truth optical flow field is shown in **5**, the occlusions estimated with occlusion check in **6**. Due to almost full occlusion of the area marked by the red ellipse in **1**, matches for the corresponding boundary segment are wrong. The boundary is mistakenly treated as a motion boundary. Additionally, the matches from deep matching in the area above the boundary are wrong and disconnecting it by the boundary leads to an error in the estimated optical flow field visualized in **8**. This error is not present in **7**, where the motion boundary detector is not used.

Chapter 4

Conclusions

This project presented an approach for extracting initial motion boundary clues from images by performing global sparse descriptor matching along image boundaries. The extracted motion boundaries were then used to enhance optical flow estimation. Using the proposed approach together with occlusion checks gives a significant performance improvement. On the selected cases, the accuracy of optical flow estimation increased by 10% (compared to standard LDOF with matches from deep matching).

In most of the cases the approach tends to underestimate the true motion boundaries and therefore does not generate false positives. However, in the case of occlusions correct matches are impossible to determine and false positives do occur. Other limitations of the approach include its inability to adequately handle homogeneous areas and the susceptibility to errors of boundary estimates produced by the Gb algorithm. These problems need to be addressed in future work.

Bibliography

[1] Brox, T., C. Bregler, and J. Malik (2009). Large displacement optical flow. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*.

[2] Brox, T. and J. Malik (2011). Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*(3), 500–513.

[3] Leordeanu, M., R. Sukthankar, and C. Sminchisescu (2012). Efficient closed-form solution to generalized boundary detection. In *European Conference on Computer Vision (ECCV)*.

[4] Weinzaepfel, P., J. Revaud, Z. Harchaoui, and C. Schmid (2013). Deepflow: Large displacement optical flow with deep matching. In *IEEE Intenational Conference on Computer Vision (ICCV)*.