Chapter 2
Optical Flow Estimation

Abstract In this chapter we review the estimation of the two-dimensional apparent motion field of two consecutive images in an image sequence. This apparent motion field is referred to as optical flow field, a two-dimensional vector field on the image plane. Because it is nearly impossible to cover the vast amount of approaches in the literature, in this chapter we set the focus on energy minimization approaches which estimate a dense flow field. The term *dense* refers to the fact that a flow vector is assigned to every (non-occluded) image pixel. Most dense approaches are based on the variational formulation of the optical flow problem, firstly suggested by Horn and Schunk. Depending on the application, density might be one important property besides accuracy and robustness. In many cases computational speed and real-time capability is a crucial issue. In this chapter we therefore discuss the latest progress in accuracy, robustness and real-time capability of dense optical flow algorithms.

Space is a still of time, while time is space in motion.
Christopher R. Hallpike

2.1 Optical Flow and Optical Aperture

This chapter is about optical flow and its estimation from image sequences. Before we go into details on optical flow estimation, we review the definition of optical flow and discuss some of the basic challenges of optical flow estimation.

DOI 10.1007/978-0-85729-965-9_2, © Springer-Verlag London Limited 2011

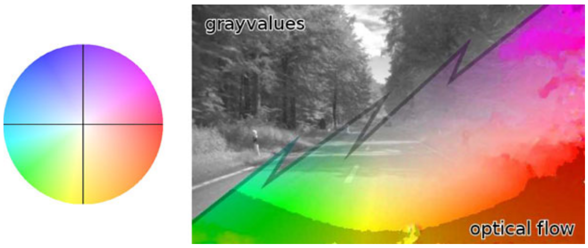

Fig. 2.1 Color coding of flow vectors: Direction is coded by hue, length by saturation. The example on the *right* shows the expanding flow field of a forward motion. Flow vectors above 20 px are saturated and appear in *darker* colors

Whenever a camera records a scene over time, the resulting image sequence can be considered as a function $I(x,y,t)$ of the gray value at image pixel position $\mathbf{x} = (x,y)^\top$ and time t. In this book we will limit ourselves to gray value sequences. Color images therefore have to be transformed accordingly, depending on the color scheme used. Furthermore, some approaches easily generalize to color images. If the camera, or an object, moves within the scene, this motion results in a time-dependent displacement of the gray values in the image sequence. The resulting two-dimensional apparent motion field in the image domain is called the optical flow field. Figure 2.1 shows the color scheme used to display such a flow vector field in this book. An example for a scene from the Middlebury optical flow benchmark with overlaid displacement vectors can be seen in Fig. 2.2.

The most common assumption used in optical flow estimation is the brightness constancy assumption. It states that the gray value of corresponding pixels in the two consecutive frames should be the same. Unfortunately, not every object motion yields a change in gray values and not every change in gray values is generated by body motion. Counter examples to this brightness constancy assumption may arise through changes in illumination or upon observations of transparent or non-Lambertian material. Another source of gray value changes is the inherent noise of the camera sensor. Especially in bad illumination conditions (e.g. at night time) the number of photons, which impact an image pixel, may vary over time. In such cases brightness constancy is violated and thus respective motion estimates will degrade.

Another source of ambiguity arises from the so-called aperture problem. The aperture problem arises as a consequence of motion ambiguity when an object is viewed through an aperture, as demonstrated in Fig. 2.3. If an untextured object moves within the image, the motion within the object area cannot be recovered

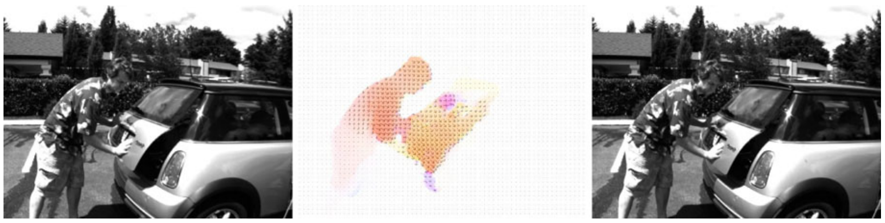

Fig. 2.2 Optical flow for the *mini cooper* scene of the Middlebury optical flow benchmark. The displacement of the image pixels is shown as displacement vectors on top of the flow color scheme defined in Fig. 2.1

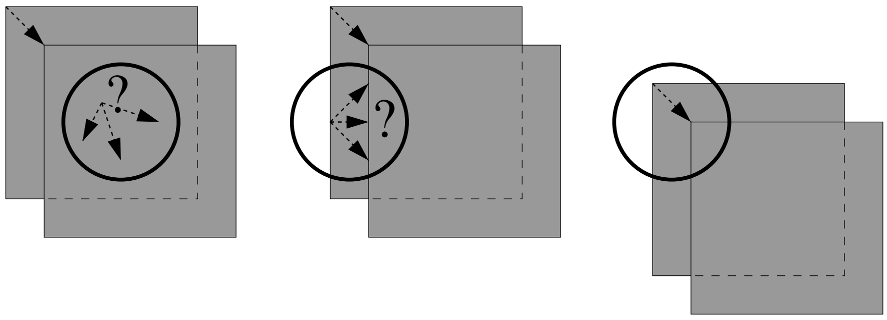

Fig. 2.3 Illustration of the aperture problem as a consequence of motion ambiguity. If an untextured object (*square*) moves within the image and is viewed through an aperture (*circle*), motion can only be recovered in the direction perpendicular to an edge. Two-dimensional motion estimation (*right*) is only possible with structural information, such as edges, in linear independent directions

without additional information. Even at object edges, where the gray value of the background differs from the gray value of the object (or object edge) itself, without additional information motion can only be recovered in one dimension. The two-dimensional motion can only be recovered where more information is available, for example at object corners or if texture is available. As a result of the aperture problem the optical flow problem, like many other inverse problems, is not well-posed in the sense of Hadamard. Thus, the optical problem needs to be re-formulated for numerical treatment. Typically this involves considering additional assumptions, such as smoothness of the optical flow field and known as regularization.

Depending on the type of regularization, optical flow algorithms are often categorized into the two general classes feature-based approaches and variational approaches. Feature-based approaches compute the optical flow displacement for a pixel and its neighborhood independently of the optical flow solutions obtained at the other pixels in the image. Section 2.2 covers feature-based approaches. Variational approaches take into account the optical flow solutions of neighboring pixels and imply smoothness assumptions on the optical flow field. Variational approaches are covered in Sect. 2.3. Variations of variational optical flow exhibit the currently best results on standard evaluation benchmarks. Section 2.4 then presents the novel Flow Refinement Framework based on an iterative improvement of a prior optical flow vector field as a third class of optical flow algorithms.

2.2 Feature-Based Optical Flow Approaches

Feature-based optical flow algorithms can again be categorized into pixel accurate optical flow approaches and sub-pixel accurate optical flow approaches. Pixel accurate optical flow approaches assign pixels of the input image to pixels of the output image. This is usually done by evaluating a pixel matching score based on gray values of a pixel's neighborhood. Algorithms of this class can be understood as a combinatorial task, because the number of possible flow vectors or displacement vectors for a certain pixel is bounded by the image size. Due to the combinatorial structure, pixel accurate optical flow algorithms can be parallelized yielding real-time efficiency on dedicated hardware. They have the nice property of robustness due to the restricted solution space (only discrete integer pixel positions are considered and outliers are less likely), but at the same time suffer from accuracy limitations, i.e. the displacements are only pixel-discrete. In Sect. 2.2.1 a census based optical flow algorithm is reviewed as a representative of this class.

Section 2.2.2 introduces the optical flow constraint, which is employed by most sub-pixel accurate optical flow algorithms. It assumes that the gray value image sequence has a continuous domain. The change of a pixel's gray value between two time instances can then be computed evaluating the image gradient. The early work of Lucas and Kanade [54] is reviewed which employs the linearized version of the optical flow constraint.

2.2.1 Census Based Optical Flow

The Census transformation is a form of non-parametric local mapping from intensity values of pixels to a signature bit string. It relies on the relative ordering of intensities in a local window surrounding a pixel and not on the intensity values themselves. Thus, it captures the image structure [118]. Applied to a local 3×3 neighborhood of an image pixel, the Census transform compares the center pixel $\mathbf{x} = (x, y)^\top$ to all other pixels $\mathbf{x}'$ inside the patch resulting in

$$\theta(I, \mathbf{x}, \mathbf{x}') = \begin{cases} 0 & \text{if } I(\mathbf{x}) - I(\mathbf{x}') > c, \\ 1 & \text{if } |I(\mathbf{x}) - I(\mathbf{x}')| \le c, \\ 2 & \text{if } I(\mathbf{x}) - I(\mathbf{x}') < -c, \end{cases}$$

with the gray value intensity $I(\mathbf{x})$ at pixel position $\mathbf{x}$. In [48, 85], the threshold c is chosen between 12 and 16 for images with gray values between 0 and 255. The Census digit θ measures the similarity between the gray values at pixel positions $\mathbf{x}$ and $\mathbf{x}'$. All census digits of the image patch are unrolled clockwise, building the signature bit string or signature vector:

$$\underset{\text{gray values}}{\begin{array}{|c|c|c|}\hline 124 & 74 & 32 \\ \hline 124 & 64 & 18 \\ \hline 157 & 116 & 84 \\ \hline \end{array}} \rightarrow \underset{\text{Census digits}}{\begin{array}{|c|c|c|}\hline 2 & 1 & 0 \\ \hline 2 & X & 0 \\ \hline 2 & 2 & 2 \\ \hline \end{array}} \rightarrow \underset{\text{signature vector}}{21002222}$$

The larger the threshold value c is, the more insensitive the result becomes to non-additive illumination changes. On the other hand, a large threshold value yields less unique signature bit strings. The signature vector is then used to search for corresponding pixel pairs.

To this end, all signature vectors of the first image are stored in a hash-table together with their pixel position. Then, all signature vectors of the second image are compared to the hash-table entries. This gives a list of putative correspondences (hypotheses) for each signature. The list is empty if a signature in the second image does not exist in the first image. In the event of multiple entries, the list can be reduced by applying photometric and geometric constraints; if there are still multiple entries, the shortest displacement vector wins (see [85] for further details).

Thanks to the indexing scheme, arbitrarily large displacements are allowed. Even if an image patch moves from the top left image corner to the bottom right corner it can potentially be correctly matched. The method has been successfully implemented on parallel hardware, allowing for real-time computation. A disadvantage of the Census based optical flow method is its pixel-accuracy; the results suffer from discretization artifacts. Furthermore, low contrast information (gray value difference below the threshold c) is ignored.

2.2.2 The Optical Flow Constraint

Most sub-pixel accurate solutions to optical flow estimation are based on the (linearized) optical flow constraint (see optical flow evaluation in [6]). The optical flow constraint states that the gray value of a moving pixel stays constant over time, i.e.

$$I(x,y,t) = I(x+u, y+v, t+1)$$

where $\mathbf{u} = (u,v)$ is the optical flow vector of a pixel $\mathbf{x} = (x,y)$ from time t to time $t+1$. The linearized version of the image function (using the first-order Taylor approximation) reads

$$\begin{aligned} I(x,y,t) &\approx I(x,y,t+1) + \nabla I(x,y,t+1)^\top \begin{pmatrix} u \\ v \end{pmatrix}, \\ 0 &= \underbrace{I(x,y,t+1) - I(x,y,t)}_{I_t(x,y,t+1)} + \nabla I(x,y,t+1)^\top \begin{pmatrix} u \\ v \end{pmatrix}. \end{aligned} \tag{2.1}$$

From now on the partial derivatives of the image function are denoted as I_t, I_x, and I_y and also the inherent dependency on the image position (x,y) is dropped yielding the optical flow constraint equation

$$\text{OFC}(u,v): \quad 0 = I_t + I_x u + I_y v. \tag{2.2}$$

The optical flow constraint has one inherent problem: it yields only one constraint to solve for two variables. It is well known that such an under-determined equation system yields an infinite number of solutions. For every fixed u a valid v can be found fulfilling the constraint. Again, the aperture problem previously discussed in Sect. 2.1 becomes visible.

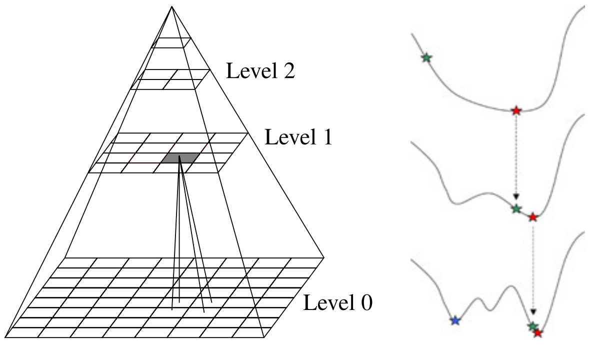

Fig. 2.4 The original image corresponds to level 0 of the image pyramid. The upper pyramid levels are down-sampled versions of the image with lower resolution; hence, high frequency image details are filtered out (*left*). For pyramid approaches, the solution on an upper pyramid level is propagated onto the lower levels and refined using the high frequency details (*right*). Figure courtesy of Thomas Brox

2.2.3 Lucas–Kanade Method

A common workaround to solve this ambiguity is to assume a constant (or affine) optical flow field in a small neighborhood of a pixel $\mathbf{x}$. Such a neighborhood $\mathcal{N}$ typically consists of $n \times n$ pixels with n smaller than 15. The optical flow constraint is then evaluated with respect to all pixels within this neighborhood window $\mathcal{N}$. This results now in an over-determined equation system because there are more equations than unknowns. In other words, in a general setting there exists no direct solution. Minimizing the sum of quadratic deviations yields the least square approach, first proposed by Lucas and Kanade in 1981 [54]:

$$\min_{u,v} \left\{ \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \big(I_t(\mathbf{x}') + I_x(\mathbf{x}')u + I_y(\mathbf{x}')v \big)^2 \right\}. \tag{2.3}$$

Extensions of this approach have been proposed. These include replacing the sum of squared errors with an absolute error measurement [5] or using Kalman filters to further gain robustness over time [73].

The resulting flow vector is sub-pixel accurate. But due to the Taylor approximation in the optical flow constraint, the method is only valid for small displacement vectors where the effect of higher order terms in (2.1) is negligible [5]. Essentially, the Lucas–Kanade method implements a gradient descent method, minimizing the deviation of the gray value between an image pixel and the gray value of its corresponding pixel (the pixel that the flow vector points to). Due to the first-order Taylor approximation consecutive executions of (2.3) and an embedding within a multi-level approach might be necessary to receive the desired result. A common approach is the use of image pyramids, solving for low frequency structures in low resolution images first and refining the search on higher resolved images (see Fig. 2.4 and [11, 60]).

One way to measure the quality of the resulting flow vector is using the sum of the gray value differences within the neighborhood $\mathcal{N}$. If this value is larger than a predefined threshold, it is likely that gradient descent was unsuccessful and either more iterations are needed or the descent process got stuck in a local minimum. While the maximum successfully matched track length depends on image content, generally speaking, flow vectors with large displacements are less likely to be found than those within a few pixels displacement. More insight into this effect can be found in [5].

The Lucas and Kanade method is currently beyond real time if a flow vector is estimated for every pixel of the input image using a common CPU implementation. Hence, Tomasi proposed to evaluate only those regions which yield a well-conditioned equation system when solving (2.3) and proposed an algorithm to search the image for such regions [92]. Recent GPU based implementations are able to track up to 10,000 image points at frame rates of 25 Hz [120]. Such KLT trackers are frequently used in applications ranging from video analysis, over structure and motion estimation to driver assistance systems.

2.3 Variational Methods

Simultaneously with the work of Lucas and Kanade, in 1981 Horn and Schunck [42] proposed another approach to cope with the under-determined optical flow constraint. The authors reverted to regularization, or smoothness, for the resulting flow field. Such smoothness was introduced by penalizing the derivative of the optical flow field, yielding an energy which was minimized by means of variational approaches:

$$\min_{u(\mathbf{x}),v(\mathbf{x})}\left\{\int_{\Omega}\left(\left|\nabla u(\mathbf{x})\right|^2+\left|\nabla v(\mathbf{x})\right|^2\right)\mathrm{d}\Omega+\lambda\int_{\Omega}\left(I_t+I_x u(\mathbf{x})+I_y v(\mathbf{x})\right)^2\mathrm{d}\Omega\right\}. \tag{2.4}$$

Here, $u(\mathbf{x})\in\mathbb{R}$ and $v(\mathbf{x})\in\mathbb{R}$ is the two-dimensional displacement vector for an image pixel $\mathbf{x}\in\mathbb{R}^2$; Ω is the image domain. The first integral (regularization term) penalizes high variations in the optical flow field to obtain smooth displacement fields. The second integral (data term) imposes the optical flow constraint (2.2) with a quadratic cost. The free parameter λ weights between the optical flow constraint and the regularization force. Variational optical flow approaches compute the optical flow field for all pixels within the image, hence they result in a dense optical flow field.

Being computationally more expensive, variational approaches only recently became more popular as processor speed has increased, allowing for the computation of dense flow fields in real time. In [18, 19], a highly efficient multi-grid approach on image pyramids is employed to obtain real-time performance. In [119] a duality-based method was proposed, which uses the parallel computing power of a GPU to gain real-time performance.

TV: $\int |\nabla f(x)|\,\mathrm{d}x$

Quadratic: $\int |\nabla f(x)|^2\,\mathrm{d}x$

Function	TV	Quadratic
$f_1(x)$	1.0	1.0
$f_2(x)$	1.0	0.1
$f_3(x)$	1.0	0.01

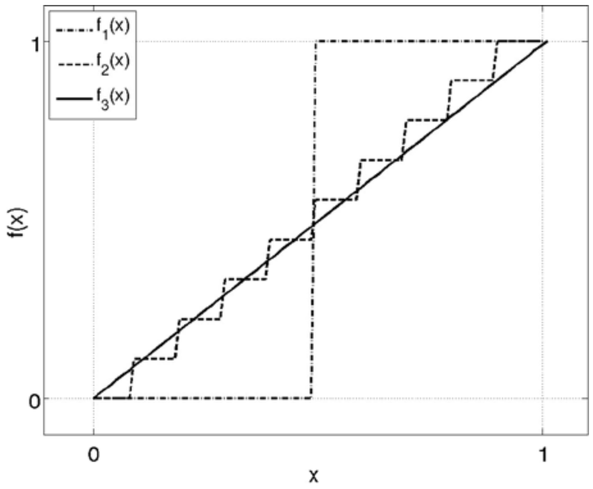

Fig. 2.5 For all three functions shown *on the right*, the total variation is identical as it merely measures the size of the total jump independent of the number of steps (1 step, 10 steps, and 100 steps, respectively). Figure courtesy of Thomas Pock [70]

The original work of Horn and Schunck suffers from the fact that its smoothness term does not allow for discontinuities in the optical flow field and the data term does not handle outliers robustly. Since discontinuities in the optical flow often appear in conjunction with high image gradients, several authors replace the homogeneous regularization in the Horn–Schunck model with an anisotropic diffusion approach [66, 113]. Others substitute the squared penalty functions in the Horn–Schunck model with more robust variants. To date the work of Horn and Schunck has attracted over 3,900 citations, many of these dealing with applications of motion estimation in different scenarios, many suggesting alternative cost functionals, and many investigating alternative minimization strategies. Cohen [24] as well as Black and Anandan [9] apply estimators from robust statistics and obtain a robust and discontinuity-preserving formulation for the optical flow energy. Aubert et al. [2] analyze energy functionals for optical flow incorporating an L^1 norm for the data fidelity term and a general class of discontinuity-preserving regularization forces. Brox et al. [13] employ a differentiable approximation of the L^1 norm for both the data and smoothness term and formulate a nested iteration scheme to compute the displacement field.

The integral of the L^1 norm of the gradient, i.e. $\int |\nabla f(x)|\,\mathrm{d}x$, is referred to as the total variation (TV norm) of f. Essentially, the integral *counts* the amount of variation, or *fluctuation*, in the data. In contrast to the original quadratic L^2-regularity suggested by Horn and Schunck, the L^1-regularity is known to better preserve discontinuities [9, 13, 30, 66, 81]. Figure 2.5 illustrates the advantage of using the L^1 norm for denoising. For any monotone function growing from 0 to 1, its total variation is 1 regardless of whether the function undergoes a gradual transition or a sharp jump. As a consequence, in contrast to the quadratic regularization proposed by Horn and Schunck, the total variation no longer favors smooth transitions over sharp transitions. In the PDE community it is therefore called "discontinuity-preserving". The total variation plays a prominent role in image analysis since it is the most discontinuity-preserving among all convex regularizers. For a review of total variation methods in image analysis we refer the reader to [22].

Zach et al. [119] proposed to solve the robust TV optical flow formulation by rewriting it into a convex dual form, thereby allowing a quadratic decoupling of the data term and the smoothness term. Let us replicate this approach in Sect. 2.3.1 and draw conclusions about the algorithmic design behind the mathematical notations. It will turn out that basically two steps are performed alternatingly, (1) data term evaluation and (2) flow field denoising.

2.3.1 Total Variation Optical Flow

Recall that the general Horn and Schunck energy (equation (2.4)) for a two-dimensional flow field (u, v) (dropping the inherent dependency on $\mathbf{x}$) is given by

$$\int_\Omega \left(|\nabla u|^2 + |\nabla v|^2 + \lambda |I_t + I_x u + I_y v|^2\right) \mathrm{d}\Omega.$$

The quadratic regularizer is surprisingly simple, favoring flow fields which are spatially smooth. In contrast to the original L_2-regularity suggested by Horn and Schunck, the L_1-regularity is known to better preserve discontinuities [9, 30, 64, 67, 81]. In recent years, researchers have suggested far more sophisticated regularization techniques based on statistical learning [90]. So far these have not been able to outperform the more naive approaches. Of course it is hard to say why this is the case, one reason may be that the challenge of learning "typical" flow patterns may not be feasible, given that different image structures, unknown object deformations and camera motions may give rise to a multitude of motion patterns with little resemblance between motion fields from different videos. Nevertheless, appropriate regularization is of utmost importance for optic flow estimation, since it stabilizes the otherwise ill-posed optical flow problem and induces a filling-in in areas of low contrast or texture.

Replacing the quadratic penalizers with robust versions (i.e. the absolute function) yields

$$\int_\Omega \left(|\nabla u| + |\nabla v| + \lambda |I_t + I_x u + I_y v|\right) \mathrm{d}\Omega. \tag{2.5}$$

Although (2.5) seems to be simple, it offers computational difficulties. The main reason is that the regularization term and the data term are not continuously differentiable. One approach is to replace the absolute function $|x|$ with the differentiable approximation $\Psi_\varepsilon(x) = \sqrt{x^2 + \varepsilon^2}$, and to apply a numerical optimization technique on this slightly modified functional (e.g. [13, 23]). In [119] the authors propose an exact numerical scheme to solve (2.5) by adopting the *Rudin–Osher–Fatemi* (ROF) energy [78] for total variation based image denoising to optical flow. We will review their approach in this section. The minimization of the ROF energy for image denoising will be covered in more detail in Sect. 2.4.

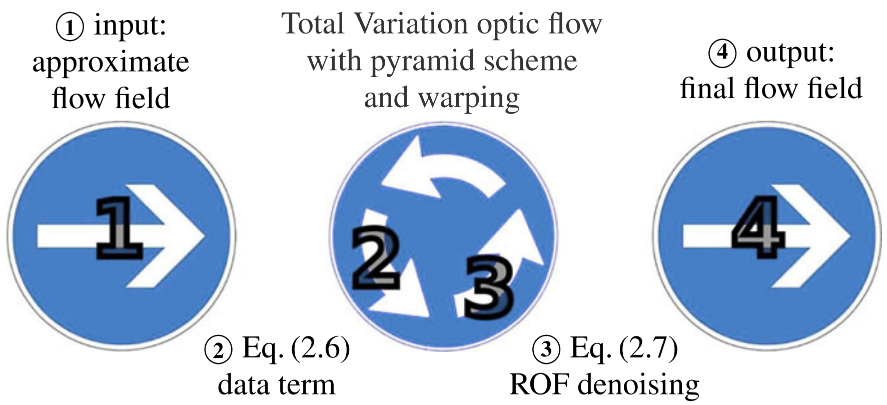
① input: approximate flow field
Total Variation optic flow with pyramid scheme and warping
④ output: final flow field
1
2
3
4
② Eq. (2.6) data term
③ Eq. (2.7) ROF denoising

Fig. 2.6 Basic algorithmic design of the total variation optical flow algorithm. Subsequent data term evaluation and denoising steps, embedded into a warping strategy, are applied to an approximate flow field (e.g. starting with the zero flow field) to compute the resulting flow field

2.3.2 *Quadratic Relaxation*

Inspired by algorithms for TV-L1 minimization like the one in [3], Zach and coworkers [119] introduce auxiliary (so-called dual) variables u' and v' and replace one instance of the original variables u and v in (2.5) with the dual variables, yielding

$$\min_{u,v,u',v'} \left\{ \int_\Omega \left(|\nabla u| + |\nabla v| + \lambda |I_t + I_x u' + I_y v'| \right) \mathrm{d}\Omega \right\} \quad \text{with } u = u' \text{ and } v = v'.$$

This step does not change the energy function. The key novelty is that the side conditions $u = u'$ and $v = v'$ are now relaxed and embedded into the energy functional to be minimized. This *quadratic relaxation* yields the following *strictly convex* approximation of the original total variation regularized optical flow problem

$$\min_{u,v,u',v'} \left\{ \int_\Omega \left(|\nabla u| + |\nabla v| + \frac{1}{2\theta}(u-u')^2 + \frac{1}{2\theta}(v-v')^2 \right.\right. \\ \left.\left. + \lambda |I_t + I_x u' + I_y v'| \right) \mathrm{d}\Omega \right\}, \tag{2.6}$$

where θ is a small constant, such that u is a close approximation of u' (and equivalently v is a close approximation of v'). Due to the quadratic germs, the resulting energy is strictly convex, hence it has a unique minimum. This minimum is found by alternating steps updating either the dual variables, u' and v', or the primal variables, u and v, in every iteration (see also Fig. 2.6). More specifically the two steps are

1. For u and v being fixed, solve

$$\min_{u',v'} \int_\Omega \left\{ \frac{1}{2\theta}(u-u')^2 + \frac{1}{2\theta}(v-v')^2 + \lambda |I_t + I_x u' + I_y v'| \right\} \mathrm{d}\Omega. \tag{2.7}$$

Note that this minimization problem can be independently evaluated for every pixel because no spatial derivatives of the flow field contribute to the energy. More specifically, the optical flow constraint data term, $\lambda|I_t + I_x u' + I_y v'|$, is minimized w.r.t. the dual flow variables u' and v' such that deviations from the primal variables are penalized quadratically. This *demands a solution close to the given (approximate) primal flow field.*

2. For u' and v' being fixed, solve

$$\min_{u,v} \int_{\Omega} \left\{ |\nabla u| + |\nabla v| + \frac{1}{2\theta}(u - u')^2 + \frac{1}{2\theta}(v - v')^2 \right\} \mathrm{d}\Omega. \tag{2.8}$$

This is the total variation based image denoising model of Rudin, Osher, and Fatemi [78]. The denoising will be presented in more detail within the general framework in Sect. 2.4. For now, note that *a denoising of the dual variables, u' and v'*, is *computed.*

Embedding (2.7) and (2.8) into a pyramid warping scheme implies that the denoised solution vector (u, v) serves as a new approximate flow field (u, v) for the next iteration or next lower pyramid level. Iteratively solving the data term and subsequent denoising yields the final optical flow result.

This basic idea of data term solving and denoising is picked up in Sect. 2.4 and a generalized approach is presented which is called *Refinement Optical Flow*. In that section the solutions for the minimization problems (2.7) and (2.8) are also outlined. But beforehand, let us conclude the literature overview on optical flow approaches.

2.3.3 Large Displacement Flow: Novel Algorithmic Approaches

Traditionally large displacements in the flow can be captured despite the data term linearization by iteratively solving the optic flow problem in a coarse-to-fine manner, where on each level of the pyramid one of the two images is being warped according to the flow field estimated on the coarser scale. Clearly, this strategy is suboptimal because fine-scale structures of low contrast may easily disappear on the coarser scales and hence will not be recovered in the final flow field.

There have been efforts to improve flow estimation for the case of large displacement: One such method proposed by Brox et al. [12] is based on imposing SIFT feature correspondence as a constraint in the optic flow estimation, thereby driving the estimated flow field to match corresponding features. While this may fail due to incorrect correspondences, in cases where SIFT correspondence is reliable it will help to capture large-displacement flow. On the other hand, the approach is still based on the traditional coarse-to-fine hierarchical estimation scheme.

From an optimization point of view the limitations regarding the estimation of large-displacement flow invariably arise because the original (non-linearized) optic flow functional is not convex.

Interestingly the above quadratic relaxation approach can be exploited even further, giving rise to a method proposed by Steinbruecker et al. [86] which can capture

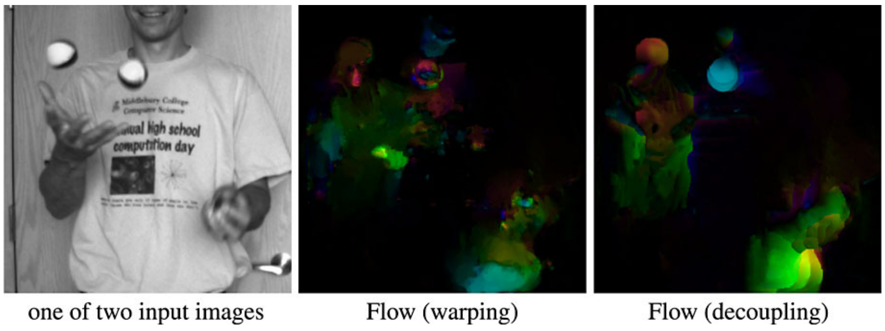

Fig. 2.7 One of two images from the Middlebury data set, the flow field computed using traditional warping and the flow field computed the decoupling approach in [86]. The latter approach allows to better capture large displacements of small scale structures like the balls

large-displacement flow without reverting to a coarse-to-fine warping strategy. The key idea is to consider the quadratic relaxation of (2.6), but for the non-linearized optic flow problem:

$$\int_{\Omega} |\nabla u| + |\nabla v| + \frac{1}{2\theta}(u-u')^2 + \frac{1}{2\theta}(v-v')^2 + \lambda \left| I_1(x) - I_2\left(x + \begin{pmatrix} u' \\ v' \end{pmatrix}\right)\right| \mathrm{d}\Omega. \tag{2.9}$$

A closer look reveals that the two optimization problems in (u, v) and in (u', v') can each be solved globally: The optimization in (u, v) for fixed (u', v') is the convex ROF problem and can therefore be solved globally. And the optimization in (u', v') for fixed (u, v) is a pointwise optimization problem which can be solved (in a discretized setting) by a complete search for each pixel. The latter can be drastically accelerated using parallel processors such as the GPU. One can therefore directly solve the above problem alternating the two solutions for (u, v) and (u', v'), always reducing the coupling parameter $\theta \to 0$ during iterations. In practice, this implies that the original flow estimation problem is solved by alternatingly determining best correspondences in the spatial vicinity and smoothing the correspondence field. Figure 2.7 shows a comparison of standard warping techniques with the above strategy: It shows that indeed small structures undergoing large displacement might be better captured using this technique.

While the above solution allows a decoupling of the original flow problem into two problems each of which can be minimized globally, the overall solution may still be far from being globally optimal. Nevertheless, recent efforts aim at finding convex relaxations of the non-linearized flow estimation problem [34]. The key idea is to formulate flow estimation as a problem of multilabel optimization and to devise appropriate convex representations which account for the fact that labels are vector-valued. The resulting algorithms give rise to optic flow solutions which are independent of initialization and minimization strategy. Moreover they come with a quality guarantee in terms of a computable energetic bound (of typically 5%) from the (unknown) global optimum.

2.3.4 Other Optical Flow Approaches

Over the last years, the number of optical flow algorithms with increasingly good performance has grown dramatically and it becomes difficult to summarize all contributions. Robust variational approaches have been shown to yield the most accurate results to the optical flow problem known in literature. But they cover only a subset of dense optical flow algorithms. In [91] the authors pick up recent developments in optical flow estimation and reveal some of the secrets for accurate flow estimation techniques. They observe that median filtering of optical flow values is one clue to accurate flow fields. Consequently, they propose a variational approach based on a continuous version of the median filter. We will take a closer look at the median filter later in this chapter.

In [84], a two-step method is presented where first optical flow is computed in a local neighborhood and secondly a dense flow field is derived in a relaxation step. An unbiased second-order regularizer for flow estimation was proposed in [95]. We shall revisit this approach in Sect. 2.4.2.5. Other optical flow algorithms with outstanding performance are graph cuts (discrete optimization) [26] or the fusion of previously mentioned approaches into an optimized result [53]. A similar concept of fusion in a continuous setting was proposed in [94]. For more information on recent developments and the latest state-of-the-art optical flow algorithms, we refer to the Middlebury optical flow evaluation homepage [6].

2.4 The Flow Refinement Framework

Although the optical flow constraint, (2.2), was originally formulated for every single pixel individually, it is impossible to calculate the two translational values of optical flow using only a single reference pixel. This is part of the aperture problem, the aperture size of a single pixel is too small. We have seen solutions based on extending the aperture (Census flow and the Lucas–Kanade method) and variational approaches where smoothness is used to regularize the optical flow field solution. In this chapter we discuss a third method, based on evaluating the flow field in the proximity of a given prior flow field. Figure 2.8 illustrates how this compares to the approaches revisited beforehand.

The idea of (1) minimizing the weighted sum of a prior flow field and the optical flow constraint followed by (2) smoothing the optical flow result is very similar to the ROF-denoising based total variation optical flow approach. If iteratively used, the algorithm is given by the general make-up in Fig. 2.9. This general make-up consists of a pixel-wise (local) data term optimization step ② and a global (non-local) smoothness term evaluation step ③. Both steps are presented separately in Sects. 2.4.1 and 2.4.2. Section 2.4.3 summarizes the algorithm and provides implementation details.

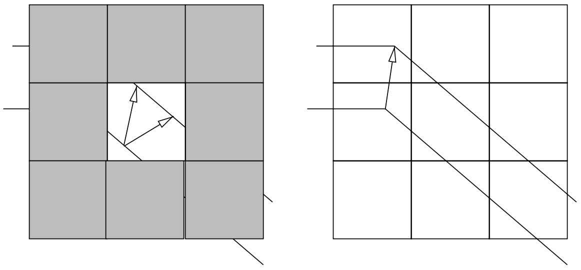

Fig. 2.8 If the aperture is too small, a motion vector cannot be computed using a single pixel (*left*). Therefore authors have used either larger apertures or a smoothing constraint, enforcing similar optical flow values for neighboring pixels (*middle*). In case a prior flow field is given, minimizing the weighted sum of the optical flow constraint and the distance to the prior solution (*right*) also yields a unique solution (resulting flow vector not shown in the illustration)

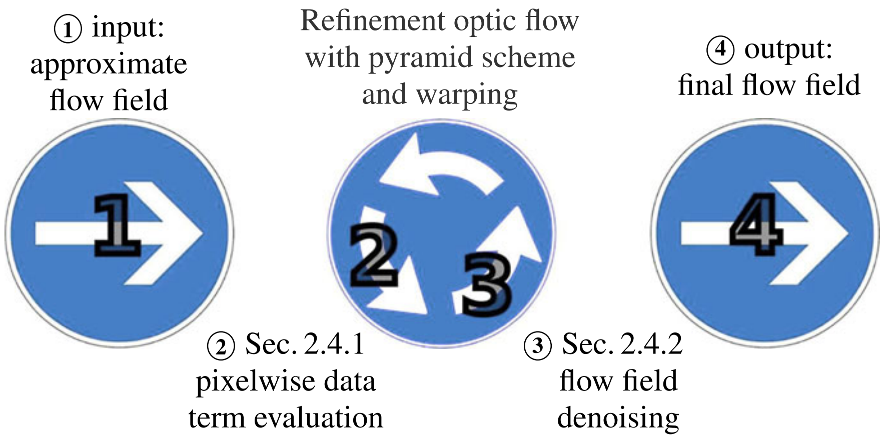
① input: approximate flow field
Refinement optic flow with pyramid scheme and warping
④ output: final flow field
1
2
3
4
② Sec. 2.4.1 pixelwise data term evaluation
③ Sec. 2.4.2 flow field denoising

Fig. 2.9 The general framework for Refinement Optical Flow. An approximate flow field is refined by iteratively evaluating the data term and subsequent smoothing, embedded in a pyramid warping scheme. The result is an optimized and refined optical flow field (compare with Fig. 2.6)

2.4.1 Data Term Optimization

The data term represents the driving force for optical flow computation. Instead of limiting ourselves to the two-dimensional (optical flow) case, we present the findings in this section using an n-dimensional flow vector. Given an approximate n-dimensional flow field $\mathbf{u}' \in \mathbb{R}^n$ as prior, an optimal solution is computed for every pixel which fulfills best a single or multiple data terms. More specifically, a solution $\mathbf{u} = (u_1, \ldots, u_n)^\top \in \mathbb{R}^n$ is obtained, which minimizes the weighted sum of the distance to the prior flow field and the individual data term violations $p(\cdot)$,

$$\mathbf{u} = \min_{\mathbf{u}} \left\{ \frac{1}{2} \|\mathbf{u} - \mathbf{u}'\|^2 + \sum_{p \in \mathcal{P}} \omega\big(\lambda_p\, p(\mathbf{u})\big) \right\}. \tag{2.10}$$

Here, $\mathcal{P}$ is the set of all data terms and each data term is given an individual weight λ_p. ω is a function penalizing deviations from 0. In this subsection, the robust L^1 norm will be used, therefore $\omega(x) = |x|$. A typical data term is the brightness constancy constraint, $p(\mathbf{u}) = I_1(\mathbf{x}) - I_2(\mathbf{x} + \mathbf{u})$. In fact, for a single data term this becomes the methodology employed in [119]. See Chap. 8 for more details on different data terms.

The investigated data terms in this subsection employ linearized constraint violations (e.g. violations of the optical flow constraint (2.2)). The linearized version of a constraint $p(\mathbf{u})$ consists of a constant scalar value, denoted by a subscript zero p_0, and a vector denoted by $\mathbf{p}$, i.e.

$$\lambda_p p(\mathbf{u}) \approx p_0 + \mathbf{p}^\top \mathbf{u} = p_0 + p_1 u_1 + \cdots + p_n u_n.$$

Note that the weight λ_p is included in the data term entries p_i for convenience reasons, where p_i is the ith component of the vector $\mathbf{p}$. In summary, in this section we will be dealing with the following data term:

$$\mathbf{u} = \min_{\mathbf{u}} \left\{ \frac{1}{2} \|\mathbf{u} - \mathbf{u}'\|^2 + \sum_{p \in \mathcal{P}} \left| p_0 + \mathbf{p}^\top \mathbf{u} \right| \right\}. \tag{2.11}$$

The remainder of this subsection deals with the optimization of (2.11) and presents a simple example employing the optical flow constraint. Although (2.11) may look quite harmless, it offers some computational difficulties. Due to the absolute function in the robust data deviation term, (2.11) is not differentiable at $p_0 + \mathbf{p}^\top \mathbf{u} = 0$. This is a pity, because the violations of the data terms should be as small as possible demanding the data terms to vanish and causing numerical instabilities. Two possible ways to solve this issue are using an ε-approximation of the absolute norm [13] and quadratic optimization techniques.

The ε-approximation of the absolute norm yields a lagged iterative solution, while quadratic optimization yields an exact solution within a fixed number of thresholding steps. In the following subsection, the lagged iterative solution is presented. Then, quadratic optimization techniques are derived for a single, two, and multiple data terms.

2.4.1.1 Approximating the Absolute Function

The absolute function $|g(x)|$ is not differentiable in $x = 0$. An often used approximation (e.g. see [13]) is to replace the absolute function by $\Psi(g(x)) = \sqrt{g(x)^2 + \varepsilon^2}$ with a small constant $\varepsilon > 0$. This function is convex and differentiable. Its derivative is given by

$$\Psi'\big(g(x)\big) = \frac{g'(x)\, g(x)}{\sqrt{g(x)^2 + \varepsilon^2}}.$$

Due to the appearance of $g(x)$ in the derivative, an iterative approach is commonly used in order to minimize the function w.r.t. $g(x)$ by gradient descent. Such a solution approach is also called *lagged feedback* because, usually, a few iterations are needed to get close to the global minimum.

Replacing the absolute functions in (2.11) with the Ψ functions yields the following approximation for the optimization problem:

$$\min_{\mathbf{u}} \left\{ \frac{1}{2} \|\mathbf{u} - \mathbf{u}'\|^2 + \sum_{p \in \mathcal{P}} \Psi\big(p_0 + \mathbf{p}^\top \mathbf{u}\big) \right\}. \tag{2.12}$$

The objective function now is convex and the minimum is found by successive gradient descent steps. This is done by setting its derivative w.r.t. $\mathbf{u}^k$, with k being the iteration index, equal zero. The starting point $\mathbf{u}^0$ is arbitrary because the objective function is convex. This yields

$$\begin{aligned} \mathbf{u}^k &= \mathbf{u}' - \sum_{p \in \mathcal{P}} \left\{ \frac{1}{\sqrt{(p_0 + \mathbf{p}^\top \mathbf{u}^{k-1})^2 + \varepsilon^2}} \mathbf{p}\big(p_0 + \mathbf{p}^\top \mathbf{u}^k\big) \right\}, \\ \mathbf{u}^k &= \left(\mathbf{Id} + \sum_{p \in \mathcal{P}} \frac{1}{\sqrt{(p_0 + \mathbf{p}^\top \mathbf{u}^{k-1})^2 + \varepsilon^2}} \mathbf{p}\mathbf{p}^\top \right)^{-1} \\ &\quad \times \left(\mathbf{u}' - \sum_{p \in \mathcal{P}} \left\{ \frac{1}{\sqrt{(p_0 + \mathbf{p}^\top \mathbf{u}^{k-1})^2 + \varepsilon^2}} p_0 \mathbf{p} \right\} \right). \end{aligned}$$

Due to the lagged feedback, in general a few iterations (re-linearizations) are needed to find a minimum which is close to the optimum.

2.4.1.2 Quadratic Optimization

Another approach to solve the objective equation, (2.11), is quadratic programming. In that case, the optimization problem involving the absolute function is transformed into a quadratic optimization problem with inequality constraints. Some fundamentals for optimization techniques based on quadratic optimization will now be presented. Finally, a fast and optimal thresholding scheme for solving the resulting quadratic optimization problem is presented.

In a first step, the absolute functions (the original data terms $p \in \mathcal{P}$) are replaced by dual variables p' and inequality constraints (see Fig. 2.10 for an example). The remaining minimization problem is to find the minimum of

$$\frac{1}{2} \|\mathbf{u} - \mathbf{u}'\|^2 + \sum_{p \in \mathcal{P}} p' \tag{2.13}$$

with inequality constraints

$$\begin{aligned} &\text{(i)} \quad p_0 + \mathbf{p}^\top \mathbf{u} - p' \le 0 \quad \forall p \in \mathcal{P} \quad \text{and} \\ &\text{(ii)} \quad -p_0 - \mathbf{p}^\top \mathbf{u} - p' \le 0 \quad \forall p \in \mathcal{P}. \end{aligned} \tag{2.14}$$

Equation (2.13) with side conditions (2.14) yields the same minimum as (2.11). The absolute function has been replaced yielding a differentiable objective function. This, however, is at the cost of one additional variable and two inequality constraints per data term. Note that (2.13) is a combination of convex functions and

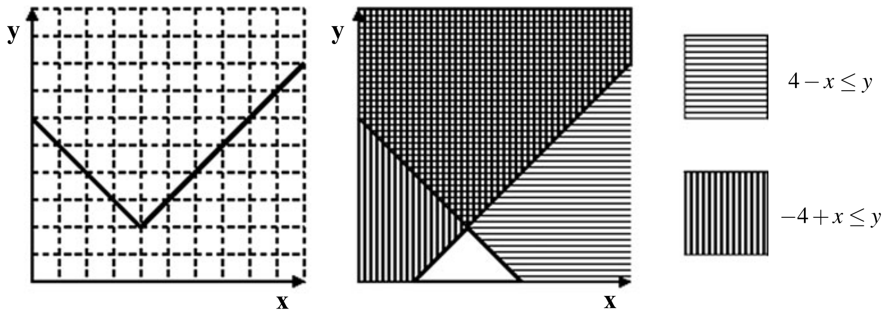

Fig. 2.10 Example for finding $\min_x\{2 + |4 - x|\}$ (minimum $x^* = 4$). The $|\cdot|$ function is replaced by a dual variable y and inequality constraints (*equations to right*), yielding $\min_{x,y}\{2 + y\}$. The minimum is $(x^*, y^*) = (4, 2)$. Note that both problems yield the same minimum for the primal variable x^*

hence is convex itself. Let us now use the Karush–Kuhn–Tucker theorem [47] for convex quadratic minimization problems under linear inequality constraints to find an optimal solution to (2.13) and hence the original problem (2.11).

Proposition 2.1 *For a convex quadratic minimization problem* $\min_x\{f(x)\}$, *under* N *linear inequality constraints* $g_i(x) \leq 0$ *where* $0 \leq i \leq N$, *a global optimum of a solution* x^* *holds true if there exist constants* μ_i *such that the Karush–Kuhn–Tucker* (*KKT*) *conditions* [47] *are fulfilled*:

(i) *Stationarity*: $\nabla f(x^*) + \sum_i \mu_i \nabla g_i(x^*) = 0,$

(ii) *Primal feasibility*: $g_i(x^*) \leq 0,$

(iii) *Dual feasibility*: $\mu_i \geq 0,$

(iv) *Complementary slackness*: $\mu_i g_i(x) = 0 \quad \forall i.$

Solution schemes for a single data term, two data terms based on thresholding, and for the general case of multiple data terms are now presented in the following.

2.4.1.3 Single Data Term

For a single data term, the task is to find the vector $\mathbf{u}^*$ which solves the minimization problem

$$\mathbf{u}^* = \min_{\mathbf{u}} \left\{ \frac{1}{2} \|\mathbf{u} - \mathbf{u}'\|^2 + \left| p_0 + \mathbf{p}^\top \mathbf{u} \right| \right\}, \tag{2.15}$$

or its dual formulation

$$\mathbf{u}^* = \min_{\mathbf{u}} \left\{ \frac{1}{2} \|\mathbf{u} - \mathbf{u}'\|^2 + p' \right\} \tag{2.16}$$

Table 2.1 Thresholding checks for a single data term

Thresholding check	Solution
$p(\mathbf{u}') < -\mathbf{p}^\top\mathbf{p}$	$\mathbf{u}^* = \mathbf{u}' + \mathbf{p}$
$\lvert p(\mathbf{u}')\rvert \le \mathbf{p}^\top\mathbf{p}$	$\mathbf{u}^* = \mathbf{u}' - \frac{p(\mathbf{u}')}{\mathbf{p}^\top\mathbf{p}}\mathbf{p}$
$p(\mathbf{u}') > \mathbf{p}^\top\mathbf{p}$	$\mathbf{u}^* = \mathbf{u}' - \mathbf{p}$

with linear side conditions

$$\begin{aligned} p_0 + \mathbf{p}^\top\mathbf{u} - p' &\le 0 \quad \text{and} \\ -p_0 - \mathbf{p}^\top\mathbf{u} - p' &\le 0. \end{aligned}$$

The solution computes as (first presented by [119]; see Sect. 8.3.2 for the proof) in Table 2.1.

2.4.1.4 Two Data Terms

In this subsection, an efficient solution scheme to minimize the objective function (2.11) or, equivalently, its dual quadratic optimization problem (2.13) with inequality constraints (2.14) for two data terms is presented. To this end, we have two data terms, $\lambda_1 p_1(\mathbf{u}) = a_0 + \mathbf{a}^\top\mathbf{u}$ and $\lambda_2 p_2(\mathbf{u}) = b_0 + \mathbf{b}^\top\mathbf{u}$, where a_0 and b_0 are constant and $\mathbf{a}$ and $\mathbf{b}$ represent the linear parts of the data terms. The minimization problem states

$$\min_{\mathbf{u}} \left\{ \frac{1}{2}\|\mathbf{u} - \mathbf{u}'\|^2 + \big|\underbrace{a_0 + \mathbf{a}^\top\mathbf{u}}_{\lambda_1 p_1(\mathbf{u})}\big| + \big|\underbrace{b_0 + \mathbf{b}^\top\mathbf{u}}_{\lambda_2 p_2(\mathbf{u})}\big| \right\}. \tag{2.17}$$

Its dual formulation with dual variables a' and b' is

$$\min_{\mathbf{u}} \left\{ \frac{1}{2}\|\mathbf{u} - \mathbf{u}'\|^2 + a' + b' \right\} \quad \text{such that} \quad \begin{array}{rl} \text{(i)} & a_0 + \mathbf{a}^\top\mathbf{u} - a' \le 0, \\ \text{(ii)} & -a_0 - \mathbf{a}^\top\mathbf{u} - a' \le 0, \\ \text{(iii)} & b_0 + \mathbf{b}^\top\mathbf{u} - b' \le 0, \\ \text{(iv)} & -b_0 - \mathbf{b}^\top\mathbf{u} - b' \le 0. \end{array} \tag{2.18}$$

The thresholding steps for solving (2.17) are given in Table 2.2; the derivation can be found in Sect. 8.3.3.

2.4.1.5 Multiple Data Terms

If more than two data terms are given, a direct solution via thresholding becomes computationally expensive due to the high number of comparisons needed. Let the number of data terms be n. Then for every possible solution (each data term may be negative, positive or equal to zero yielding 3^n possible solutions), one has to

Table 2.2 Thresholding checks for two data terms

Thresholding checks	Solution
$a_0 + \mathbf{a}^\top(\mathbf{u}' - \mathbf{a} - \mathbf{b}) \geq 0$ $b_0 + \mathbf{b}^\top(\mathbf{u}' - \mathbf{a} - \mathbf{b}) \geq 0$	$\mathbf{u}^* = \mathbf{u}' - \mathbf{a} - \mathbf{b}$
$a_0 + \mathbf{a}^\top(\mathbf{u}' - \mathbf{a} + \mathbf{b}) \geq 0$ $b_0 + \mathbf{b}^\top(\mathbf{u}' - \mathbf{a} + \mathbf{b}) \leq 0$	$\mathbf{u}^* = \mathbf{u}' - \mathbf{a} + \mathbf{b}$
$a_0 + \mathbf{a}^\top(\mathbf{u}' + \mathbf{a} - \mathbf{b}) \leq 0$ $b_0 + \mathbf{b}^\top(\mathbf{u}' + \mathbf{a} - \mathbf{b}) \geq 0$	$\mathbf{u}^* = \mathbf{u}' + \mathbf{a} - \mathbf{b}$
$a_0 + \mathbf{a}^\top(\mathbf{u}' + \mathbf{a} + \mathbf{b}) \leq 0$ $b_0 + \mathbf{b}^\top(\mathbf{u}' + \mathbf{a} + \mathbf{b}) \leq 0$	$\mathbf{u}^* = \mathbf{u}' + \mathbf{a} + \mathbf{b}$
$a_0 + \mathbf{a}^\top(\mathbf{u}' - \mathbf{a} - \frac{\mathbf{b}}{\mathbf{b}^\top\mathbf{b}}(b_0 + \mathbf{b}^\top\mathbf{u}' - \mathbf{b}^\top\mathbf{a})) \geq 0$ $\lvert\frac{1}{\mathbf{b}^\top\mathbf{b}}(b_0 + \mathbf{b}^\top\mathbf{u}' - \mathbf{b}^\top\mathbf{a})\rvert \leq 1$	$\mathbf{u}^* = \mathbf{u}' - \mathbf{a} - \frac{\mathbf{b}}{\mathbf{b}^\top\mathbf{b}}(b_0 + \mathbf{b}^\top\mathbf{u}' - \mathbf{b}^\top\mathbf{a})$
$a_0 + \mathbf{a}^\top(\mathbf{u}' + \mathbf{a} - \frac{\mathbf{b}}{\mathbf{b}^\top\mathbf{b}}(b_0 + \mathbf{b}^\top\mathbf{u}' + \mathbf{b}^\top\mathbf{a})) \leq 0$ $\lvert\frac{1}{\mathbf{b}^\top\mathbf{b}}(b_0 + \mathbf{b}^\top\mathbf{u}' + \mathbf{b}^\top\mathbf{a})\rvert \leq 1$	$\mathbf{u}^* = \mathbf{u}' + \mathbf{a} - \frac{\mathbf{b}}{\mathbf{b}^\top\mathbf{b}}(b_0 + \mathbf{b}^\top\mathbf{u}' + \mathbf{b}^\top\mathbf{a})$
$b_0 + \mathbf{b}^\top(\mathbf{u}' - \mathbf{b} - \frac{\mathbf{a}}{\mathbf{a}^\top\mathbf{a}}(a_0 + \mathbf{a}^\top\mathbf{u}' - \mathbf{a}^\top\mathbf{b})) \geq 0$ $\lvert\frac{1}{\mathbf{a}^\top\mathbf{a}}(a_0 + \mathbf{a}^\top\mathbf{u}' - \mathbf{a}^\top\mathbf{b})\rvert \leq 1$	$\mathbf{u}^* = \mathbf{u}' - \mathbf{b} - \frac{\mathbf{a}}{\mathbf{a}^\top\mathbf{a}}(a_0 + \mathbf{a}^\top\mathbf{u}' - \mathbf{a}^\top\mathbf{b})$
$b_0 + \mathbf{b}^\top(\mathbf{u}' + \mathbf{b} - \frac{\mathbf{a}}{\mathbf{a}^\top\mathbf{a}}(a_0 + \mathbf{a}^\top\mathbf{u}' + \mathbf{a}^\top\mathbf{b})) \leq 0$ $\lvert\frac{1}{\mathbf{a}^\top\mathbf{a}}(a_0 + \mathbf{a}^\top\mathbf{u}' + \mathbf{a}^\top\mathbf{b})\rvert \leq 1$	$\mathbf{u}^* = \mathbf{u}' + \mathbf{b} - \frac{\mathbf{a}}{\mathbf{a}^\top\mathbf{a}}(a_0 + \mathbf{a}^\top\mathbf{u}' + \mathbf{a}^\top\mathbf{b})$
If all checks fail	$\mathbf{u}^* = \left[\begin{smallmatrix}\mathbf{a}^\top \\ \mathbf{b}^\top\end{smallmatrix}\right]^{-1}\left[\begin{smallmatrix}-a_0 \\ -b_0\end{smallmatrix}\right]$

perform n thresholding checks. Hence, the total number of comparisons is $n \cdot 3^n$; it increases exponentially. Of course, the search for the minimum can be stopped once a solution is found, yielding $n \cdot 3^n$ only in the worst case. With one data term, three comparisons have to be performed. Two data terms lead to a worst case of 18 comparisons (see Table 2.2) and for three data terms up to 81 comparisons are necessary.

This leads to the question, whether other quadratic optimization techniques can be used to minimize the objective function. A large number of approximate algorithms are known which approximate the linear constraint functions with barrier or penalty functions [55]. An exact solution is given by the so-called *active set method* [25]. Here, the solution is found by gradient descent within an active subset of inequality conditions. However, the algorithmic complexity also prohibits a very large number of data terms. Hence, approximating the absolute function (e.g. as in (2.12)) or approximating the inequality constraints is the most practicable approach to handle multiple data terms efficiently.

Table 2.3 Thresholding checks for two toy example with one data term

Thresholding check	Solution
$I_t + I_x u'_1 + I_y u'_2 < -\lambda \lVert \nabla I \rVert^2$	$\binom{u_1}{u_2} = \binom{u'_1}{u'_2} + \lambda \nabla I$
$\lvert I_t + I_x u'_1 + I_y u'_2 \rvert \le \lambda \lVert \nabla I \rVert^2$	$\binom{u_1}{u_2} = \binom{u'_1}{u'_2} - \frac{I_t + I_x u'_1 + I_y u'_2}{\lVert \nabla I \rVert^2} \nabla I$
$I_t + I_x u'_1 + I_y u'_2 > \lambda \lVert \nabla I \rVert^2$	$\binom{u_1}{u_2} = \binom{u'_1}{u'_2} - \lambda \nabla I$

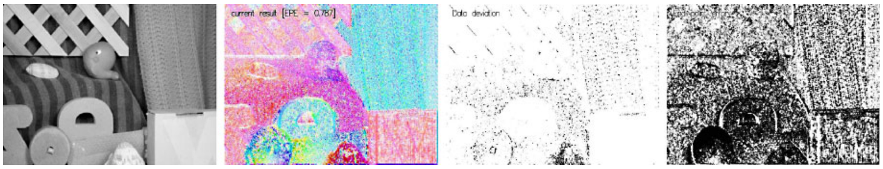

Fig. 2.11 Optical flow for the *rubber whale* sequence of the Middlebury optical flow data set using a simple thresholding step. The computed flow field (*second from left*) is noisy and shows many outliers. The data term deviation (*third from left*) is small (black corresponds to 3.75% gray value error) while the optical flow end point error is fairly large (*right image*, black denotes 0.5 px error)

2.4.1.6 Toy Example: Optical Flow Data Term

This subsection presents a toy example, minimizing (2.11) with one data term, the linearized optical flow constraint. The initial flow field to start with is set to $(u'_1, u'_2) = (0, 0)$. Ergo, small flow vectors are favored, which is correct in this very example, where most of the flow vectors have a displacement below one pixel. For the simple case of a one-channel gray value image, the data term becomes

$$\lambda p(\mathbf{u}) = \lambda(I_t + I_x u_1 + I_y u_2)$$

which yields the thresholding steps shown in Table 2.3 to solve for $u_{1,2}$ (see also [119] and Sect. 8.3.2).

As one would expect, if only the thresholding step is performed for a given initial (approximate) flow field, the resulting optical flow field shows a lot of noise. See Fig. 2.11 for such a result with $\lambda = 50$. The data term is nearly fulfilled for most part of the image while the optical flow field proves to be very noisy. Hence, it is straightforward to denoise and smooth the flow field in order to derive a more appealing solution. The next section presents methods for the second step of the Refinement Optical Flow, the smoothness term.

2.4.2 Smoothness Term Evaluation

The first step in the general framework for Refinement Optical Flow consists of a data term evaluation step. The data term optimization (2.10) is independently performed for each pixel. An advantage of such an algorithm is that it can be sped up using multiple processors and parallel computing power, e.g. modern graphics processing units (GPUs).

The disadvantage is that the solution is local in the way that only the pixel itself contributes to the solution. As seen above (compare Fig. 2.11), this leads to noisy flow fields, mainly due to three reasons; corrupted image data due to sensor noise, low entropy (information content) in the image data, and illumination artifacts.

Assuming that the noise is uncorrelated and has zero mean, the common approach to lower the noise level is a subsequent smoothing operation. However, smoothing the flow field may lead to a lack of sharp discontinuities that exist in the true flow field, especially at motion boundaries. Hence, discontinuity-preserving filters yielding a piecewise smooth flow field have to be employed.

The aperture problem (see Fig. 2.3) can only be solved if information from a region's boundary is propagated into the interior. While smoothing can be performed locally, this does not wholly solve the aperture problem (unless the filter mask is chosen large enough). Only global techniques, propagating information across the whole image, promise improvements. The three main objectives for the smoothing step can be summarized as

- Discard outliers in the flow field due to corrupted image data (denoising).
- Preserve edges, i.e. do not smooth over flow edges.
- Propagate information into areas of low texture (smoothing).

Illumination artifacts are different in nature as they cannot be removed by denoising or smoothing. Section 3.1 will tackle the illumination problem and show how to increase robustness to illumination changes by preparing the input image accordingly. In this subsection, different smoothing (or denoising) filters are presented; the median filter, total variation denoising filters, an anisotropic and edge-preserving diffusion filter, and a second-order prior denoising. Quantitative results obtained using the different filters are discussed in Sect. 3.2. In general, a filter $\mathcal{F}$ is applied to a given n-dimensional flow field $\mathbf{u}$, where the gray value image I may yield as prior; it returns the smoothed flow field $\widehat{\mathbf{u}}$ such that

$$\widehat{\mathbf{u}} = \mathcal{F}(\mathbf{u}, I). \tag{2.19}$$

2.4.2.1 Median Filtering

Discarding outliers and preserving edges is a well-known characteristic of rank filters [58]. The median filter probably is the best-known rank filter. According to [31], the median of N numerical values has the following properties:

- The median of a list of N values is found by sorting the input array in increasing order, and taking the middle value.
- The median of a list of N values has the property that in the list there are as many greater as smaller values than this element.
- The median of a list of N values minimizes the sum of deviations over all list entries.

The last property makes the median filter especially suitable for denoising if noise characteristics are unknown. A median filter has also the nice property of converging to a periodic solution if recursively executed on an input image. This periodic (for most part of the image stationary) image is called the root image of the median filter [116]. For recursively applying n median filter steps, denoted by a superscript, i.e.

$$\widehat{\mathbf{u}} = \mathcal{MF}^n(\mathbf{u}) = \mathcal{MF}\big(\mathcal{MF}^{n-1}(\mathbf{u})\big), \tag{2.20}$$

the root image property yields

$$\exists n > 0, i > 0: \quad \mathcal{MF}^{n+i}(\mathbf{u}) = \mathcal{MF}^n(\mathbf{u}).$$

Recursively applying the median filter propagates information across the image and produces piecewise smooth flow fields. For the root image, the above definitions of the median filter are inherently fulfilled; outliers are replaced by local medians, such that deviations between neighboring pixels are minimized.

In the experiments, a fixed number of median filter iterations is used. Using the *Bubble-Sort* algorithm, median filtering can be implemented quite efficiently employing a fixed number of comparisons [31]. The window size used in the experiments is 3×3. A fixed number of median filter iterations and a fixed window size are especially suitable when using parallel processing to speed up computational time.

2.4.2.2 TV-L^1 Denoising

The continuous counterpart to median filtering is total variation denoising with an L^1 data term, which provides an edge-preserving and smooth flow field. The classical TV denoising algorithm is described by Rudin, Osher, and Fatemi in [78]. Their algorithm seeks an equilibrium state (minimal energy) of an energy functional consisting of the TV norm of the data field, $|\nabla\widehat{\mathbf{u}}|$, and a fidelity term of the data field $\widehat{\mathbf{u}}$ to the noisy input data field $\mathbf{u}$:

$$\widehat{\mathbf{u}} = \mathcal{TV}_{L_p}(\mathbf{u}) = \min_{\widehat{\mathbf{u}}}\left\{\int_{\Omega}\left(|\nabla\widehat{\mathbf{u}}| + \frac{1}{2}\lambda|\widehat{\mathbf{u}} - \mathbf{u}|^p\right)\mathrm{d}\Omega\right\}. \tag{2.21}$$

Here, $\lambda \in \mathbb{R}$ is a scalar value controlling the fidelity of the solution to the input image (inversely proportional to the measure of denoising); $p = 2$ yields a quadratic penalizer and $p = 1$ linear penalizing. The resulting data field contains the *cartoon part* of the image [117]. The cartoon has a curve of discontinuities, but elsewhere it is assumed to have small or null gradient, $|\nabla\widehat{\mathbf{u}}| \approx 0$. In the original work [78], a quadratic data fidelity term ($p = 2$) was used. A closer approximation of the discrete median filter is an absolute data fidelity term [7]. The following solution scheme for total variation with an absolute data term is found in [70]:

Proposition 2.2 *The solution of*

$$\mathcal{TV}_{L_1}(\mathbf{u}) = \min_{\widehat{\mathbf{u}}}\int_{\Omega}|\nabla\widehat{\mathbf{u}}| + \frac{1}{2}\lambda|\widehat{\mathbf{u}} - \mathbf{u}|\,\mathrm{d}\Omega, \tag{2.22}$$

is given by

$$\widehat{\mathbf{u}} = \mathbf{q} + \frac{1}{2}\lambda\, \mathrm{div}\, \mathbf{p}.$$

The dual variables $\mathbf{p} = [p_1, p_2]$ *and* $\mathbf{q}$ *are defined iteratively* (*superscript denotes the iteration index, subscripts denote the pixel location*) *by computing*

$$\widetilde{\mathbf{p}}^{n+1} = \mathbf{p}^n + \frac{2\tau}{\lambda}\nabla\left(\mathbf{q}^n + \frac{1}{2}\lambda\, \mathrm{div}\, \mathbf{p}^n\right), \quad \textit{followed by } \mathbf{p}^{n+1} = \frac{\widetilde{\mathbf{p}}^{n+1}}{\max\{1, |\widetilde{\mathbf{p}}^{n+1}|\}}$$

and

$$q^{n+1}_{i,j} = \begin{cases} q^n_{i,j} - \frac{1}{2}\lambda\theta & \textit{if } q^n_{i,j} - u_{i,j} > \frac{1}{2}\lambda\theta, \\ q^n_{i,j} + \frac{1}{2}\lambda\theta & \textit{if } q^n_{i,j} - u_{i,j} < -\frac{1}{2}\lambda\theta, \\ f & \textit{otherwise} \end{cases}$$

where $\mathbf{p}^0 = \mathbf{0}$, $\mathbf{q}^0 = u$, $\theta = 0.2$ *and the time step* $\tau \leq 1/4$.

The implementation of Proposition 2.2 uses backward differences to approximate $\mathrm{div}\, \mathbf{p}$ and forward differences for the numerical gradient computation in order to have mutually adjoint operators [20]. The discrete version of the forward difference gradient $(\nabla u)_{i,j} = ((\nabla u)^1_{i,j}, (\nabla u)^2_{i,j})$ at pixel position (i, j) for a data field of width N and height M is defined as

$$(\nabla u)^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } i < N, \\ 0 & \text{if } i = N, \end{cases} \quad \text{and}$$

$$(\nabla u)^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } j < M, \\ 0 & \text{if } j = M. \end{cases}$$

The discrete version of the backward differences divergence operator is

$$\begin{aligned} (\mathrm{div}\, \mathbf{p})_{i,j} &= \left(\mathrm{div} \begin{bmatrix} p^1 \\ p^2 \end{bmatrix}\right)_{i,j} \\ &= \begin{cases} p^1_{i,j} - p^1_{i-1,j} & \text{if } 1 < i < N \\ p^1_{i,j} & \text{if } i = 1 \\ -p^1_{i-1,j} & \text{if } i = N \end{cases} + \begin{cases} p^2_{i,j} - p^2_{i,j-1} & \text{if } 1 < j < M, \\ p^2_{i,j} & \text{if } j = 1, \\ -p^2_{i,j-1} & \text{if } j = M. \end{cases} \end{aligned}$$

2.4.2.3 TV-L^2 Denoising

A solution scheme using the quadratic penalizers for (2.21), the ROF model,

$$\mathcal{TV}_{L_2}(\mathbf{u}) = \min_{\widehat{\mathbf{u}}} \int_{\Omega} |\nabla \widehat{\mathbf{u}}| + \frac{1}{2}\lambda |\widehat{\mathbf{u}} - \mathbf{u}|^2 \, \mathrm{d}\Omega,$$

was proposed in [21]:

Proposition 2.3 *The solution of*

$$\mathcal{TV}_{L_2}(\mathbf{u}) = \min_{\widehat{\mathbf{u}}} \int_{\Omega} |\nabla \widehat{\mathbf{u}}| + \frac{1}{2}\lambda(\mathbf{u} - \widehat{\mathbf{u}})^2 \, \mathrm{d}\Omega \tag{2.23}$$

is given by

$$\widehat{\mathbf{u}} = \mathbf{u} + \frac{1}{\lambda} \operatorname{div} \mathbf{p}.$$

The dual variable $\mathbf{p} = [p_1, p_2]$ *is defined iteratively with* $\mathbf{p}^0 = \mathbf{0}$, *the time step* $\tau \leq 1/4$, *and*

$$\widetilde{\mathbf{p}}^{n+1} = \mathbf{p}^n + \lambda\tau\left(\nabla\left(u + \frac{1}{\lambda}\operatorname{div}\mathbf{p}^n\right)\right) \quad \textit{and} \quad \mathbf{p}^{n+1} = \frac{\widetilde{\mathbf{p}}^{n+1}}{\max\{1, |\widetilde{\mathbf{p}}^{n+1}|\}}. \tag{2.24}$$

2.4.2.4 Structure-Adaptive Smoothing

For many image sequences, discontinuities of the motion field tend to coincide with object boundaries and discontinuities of the brightness function. Although this is certainly not always true, the quantitative experiments on the optic flow benchmark [6] will demonstrate that the introduction of brightness-adaptive smoothness leads to improvements of optic flow estimates.

An elegant theoretical treatise of image-adaptive regularization of flow fields was presented in [114]. There, the authors introduce regularizers of the form

$$\Psi\left(\nabla u^\top D(\nabla I)\, \nabla u\right) + \Psi\left(\nabla v^\top D(\nabla I)\, \nabla v\right), \tag{2.25}$$

corresponding to an inhomogeneous and potentially anisotropic regularization induced by a structure-dependent tensor $D(\nabla I)$. The central idea is that the smoothness of v along the two eigenvectors of D is weighted by the corresponding eigenvalues. In fact, anisotropic structure-dependent regularization was already proposed by Nagel in 1983 [65]. This is achieved by setting

$$D(\nabla I) = \frac{1}{|\nabla I|^2 + 2\lambda}\left(\nabla I^\perp \nabla I^{\top\perp} + \lambda^2 \,\mathrm{Id}\right)$$

where Id denotes the unit matrix. This leads to an anisotropic smoothing of $\mathbf{u}$ along the level lines of the image intensity while preserving discontinuities across level lines. Lately, a fast numerical approximation scheme for anisotropic diffusion was proposed by Felsberg in [32]. This scheme will also be evaluated in the experimental section.

In order to include structure-adaptive smoothing into the total variation denoising, an inhomogeneous isotropic regularization is considered. Following [1], discontinuities of the motion field arising at locations of strong image gradients are favored setting $D(\nabla I) = g(|\nabla I|)\,\mathrm{Id}$ with a strictly decreasing positive function

$$g\left(|\nabla I|\right) = \exp\left(-\alpha|\nabla I|^\beta\right). \tag{2.26}$$

Instead of solving (2.23) this yields the following solution scheme, proposed in [96]:

$$\mathcal{TV}_{L_2}(u, I) = \min_{\widehat{\mathbf{u}}} \int_{\Omega} g\Big(\big(|\nabla I|\big)|\nabla \widehat{\mathbf{u}}| + \frac{1}{2}\lambda(u - \widehat{\mathbf{u}})\Big)^2 \,\mathrm{d}\Omega. \tag{2.27}$$

This solution is obtained using Proposition 2.3 and replacing (2.24) with

$$\mathbf{p}^{n+1} = \frac{\widetilde{\mathbf{p}}^{n+1}}{\max\{1, \frac{|\widetilde{\mathbf{p}}^{n+1}|}{g(|\nabla I|)}\}}.$$

2.4.2.5 Advanced Priors for Motion Estimation

The total variation regularity although quite powerful for practical applications is a rather simple prior and one may ask if more sophisticated priors could be designed for the specific scenario under consideration. We will briefly sketch a few of these developments.

Total variation techniques favor spatially smooth and in particular constant and piecewise constant flow fields. This effect is not always desired. Especially if the flow field is slightly affine total variation regularization is known to cause a *staircasing effect*. As a remedy, in [95] the authors propose to penalize the variation of the second derivative. While it is rather straightforward to penalize second-order derivatives the interesting aspect in the above work is to ensure that there is no bias in the sense that all deviations from affine flows are penalized consistently. This approach is also considered in the experiments in Sect. 3.2.

While the above prior energetically favors affine flow fields, typical sequences arising in videos exhibit predominantly *rigid body motion*. The first reason is that the sequences often show a rigid scene filmed by a moving camera (a single 3D rigid motion), or rigid objects moving in a scene (multiple 3D rigid motions). In Sect. 8.2, we will detail a prior for variational flow field estimation which favors rigid flow fields. It is derived from epipolar geometry and simply penalizes deviations from motion along the epipolar lines.

There are other scenarios in which priors for flow field estimation have been designed to target a specific application. For the estimation of *fluid flows*, for example, researchers have advocated priors which emerge from physical models of fluid flow. Since such flows are governed by the Navier–Stokes equation, the priors are derived from this equation and ideally penalize deviations from the predicted dynamics—see for example [27, 79].

For now, we will continue with the toy example from Sect. 2.4.1 and show how median filtering improves the optical flow field. The example is followed by implementation details on the Refinement Optical Flow framework.

2.4.2.6 Toy Example Cont.: Denoising the Flow Field

Let us continue the toy example from Sect. 2.4.1, Fig. 2.11. The inherent problem is that the optical flow vectors after the data term evaluation are quite noisy; hence,

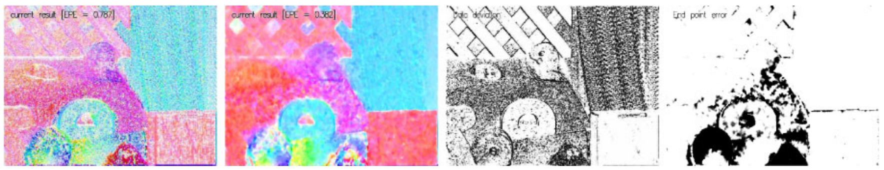

Fig. 2.12 Results after convergence of median filtering to the root image. *The leftmost image* shows the flow result from simple thresholding around the initial zero flow field. The result after convergence of iterative median filtering yields a smoother flow field and smaller flow error for most of the image (same color scale as in Fig. 2.11). This is at the cost of data term deviations

the flow field needs to be denoised. In the toy example, the median filter is applied recursively to the flow field shown in Fig. 2.11. The result can be seen in Fig. 2.12. Clearly, the flow field is much smoother than simply minimizing the data term equations while edges are preserved (compare with Fig. 2.11). The average end point error is twice as low as using a simple thresholding procedure. At the same time the data term deviation (gray value difference between current position and gray value at the end point of the flow vector) is larger for most of the image, as one would expect.

Notice that the approach in the toy example uses simple, pixel-wise thresholding and pixel-wise median filtering and can be seen as a local approach. Nevertheless, due to the iterated median filtering message passing does take place, suggesting a touch of globality. Due to the median filter, the presented method is inherently robust against a wide range of outliers. Furthermore, it benefits from the easy implementation, compared to the classical approaches of optical flow estimation (KLT [54] and Horn and Schunck [42]).

While small flow vectors are well approximated using this simple approach, large flow vectors, especially present in the lower part of the image, are still not precise. The next section reviews two common techniques around this problem: warping and image pyramids. Furthermore it provides implementation details for the involved numerical scheme. An evaluation comparing different data and smoothness terms as well as an evaluation of the Refinement Optical Flow framework on a optical flow benchmark can be found in Sect. 3.2.

2.4.3 Implementation Details

This section gives details on the implementation for the proposed Refinement Optical Flow framework. Insights on the implementation of the numerical scheme, the restriction and prolongation operators and the used warping scheme are given.

2.4.3.1 Pyramid Restriction and Prolongation

Recall that large flow vectors in Fig. 2.12, especially present in the lower part of the image, are still not precise. This is mainly due to the linearized flow constraint which does not allow for large flow vectors. Two common approaches are known to solve this problem; image pyramids and warping [13]. image pyramids use down-sampled versions of the image to find larger flow vectors whereas warping linearizes the optical flow constraint using the derived solution as a new starting point. Therefore pyramid approaches inherently use warping to propagate the results between pyramid levels.

In our experiments we use image pyramids with a scale factor of 2. Propagating results from a lower pyramid level, say 640×480 px, to a higher pyramid level, say 320×240 px is called a restriction operation. Restriction has to be performed on the gray value input images in order to assemble the image pyramid. The restriction operator is a combination of a low pass 5×5 Gaussian filter and subsequent down-sampling [87]. That is, a Gaussian filter with $\sigma = 1$,

$$\frac{1}{16}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} \times \frac{1}{16}[1\;4\;6\;4\;1] = \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix},$$

is applied to the image. Then odd rows and columns are removed from the image, yielding the lower resolved pyramid image (note that such a procedure does require the size of the input image to be a power of twice the size of the lowest resolved image).

For the opposite direction, flow vectors and dual variables have to be transformed from the higher pyramid level onto the lower pyramid level. This operation is called prolongation. The prolongation operator up-samples the image, that is, inserts odd zero rows and columns, and then applies the 5×5 Gaussian filter multiplied by 4 to it. Here, one must differentiate between up-sampling of the flow vectors $\mathbf{u}$, which have to be multiplied by an additional factor 2, and up-sampling of the dual variable $\mathbf{p}$. The dual variable $\mathbf{p}$ is not multiplied by a factor. Instead, Dirichlet boundary conditions are enforced by first setting the border of the dual variable to 0 and then up-sampling the dual variable.

Optical flow results from upper pyramid levels (e.g. 320×240 px) are a good approximation for the optical flow on lower pyramid levels (e.g. 640×480 px). Hence, they can be passed to the Refinement Optical Flow data term evaluation step as the approximate solution $\mathbf{u}'$. However, in order to allow for the estimation of large flow vectors and not to get stuck in local minima of the gray value intensity function, the optical flow constraint has to be evaluated in the vicinity of the approximate solution. This process is called warping.

2.4.3.2 Re-sampling the Coefficients of the Optical Flow Data Term via Warping

To compute the image warping, the image I_1 is linearized using the first-order Taylor approximation near $\mathbf{x} + \mathbf{u}_0$ (instead of solely $\mathbf{x}$), where $\mathbf{u}_0$ is the given (approximate)

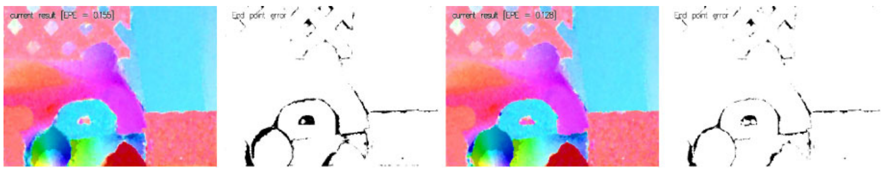

Fig. 2.13 *The two left images* show the results of the median filtered optical flow on an image pyramid. *The two right images* have been achieved using additional warps on each pyramid level. Both approaches further improve the derived optical flow field from 0.38 px average end point error (as in Fig. 2.12) to 0.155 px and 0.128 px average end point error, respectively

optical flow map:

$$I_1(\mathbf{x}+\mathbf{u}) \approx I_1(\mathbf{x}+\mathbf{u}_0) + \nabla I_1(\mathbf{x}+\mathbf{u}_0)\,(\mathbf{u}-\mathbf{u}_0).$$

The data fidelity term $\rho(\mathbf{u})$ now reads

$$\rho(\mathbf{u}) = \underbrace{I_1(\mathbf{x}+\mathbf{u}_0) - I_0(\mathbf{x}) - \nabla I_1(\mathbf{x}+\mathbf{u}_0)\mathbf{u}_0}_{c} + \nabla I_1(\mathbf{x}+\mathbf{u}_0)\mathbf{u}$$

where the left part, denoted by c, is independent of $\mathbf{u}$, and hence fixed. Commonly, bi-linear or bi-cubic look-up is used to calculate the intensity value $I_1(\mathbf{x}+\mathbf{u}_0)$ and the derivatives of I_1. The derivatives on the input images are approximated using the five-point stencil $\frac{1}{12}[-1\ 8\ 0\ -8\ 1]$. If the bi-cubic look-up falls onto or outside the original image boundary, a value of 0 is returned for the derivative and the gray value.

2.4.3.3 Toy Example Cont.: Pyramids and Warping

Figure 2.13 shows the progress in terms of optical flow end point error when applying firstly only a pyramid approach (with one initial warp for each level) and secondly additional warps on each pyramid level. Five pyramid levels and 10 warps on each pyramid level were used for this toy example. The decrease in end point error becomes visible as more sophisticated methods are used for the optical flow estimation.

2.4.3.4 Symmetric Gradients for the Data Term Evaluation

Assuming that $\mathbf{u}_0$ is a good approximation for $\mathbf{u}$, the optical flow constraint states that $I_0(\mathbf{x}) \approx I_1(\mathbf{x}+\mathbf{u}_0)$. Taking this further onto image derivatives, we obtain that $\nabla I_0(\mathbf{x})$ is a good approximation for $\nabla I_1(\mathbf{x}+\mathbf{u}_0)$. Note that replacing $\nabla I_1(\mathbf{x}+\mathbf{u}_0)$ with $\nabla I_0(\mathbf{x})$ implies that no bi-cubic look-up for the image gradients has to be employed and the computation time can be sped up. However, it turns out that using blended versions of the derivatives, larger flow vectors can be matched and hence even better results are achieved. Figure 2.14 reveals that the accuracy of the optical flow result increases (keeping all other parameters fix) if blended versions of

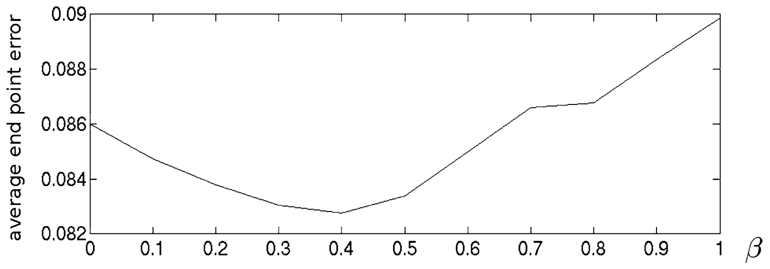

Fig. 2.14 The plot shows the optical flow accuracy, measured as the average end point error for different β values for the blending of the gradients from image I_0 and I_1 of the *rubber whale* test sequence. The improvement using a blended version of the gradients for the optical flow computation is visible

```
Input: Two intensity images $I_0$ and $I_1$
Output: Flow field $\mathbf{u}$ from $I_0$ to $I_1$
Preprocess the input images;
for $L = 0$ to max_level do
    Calculate restricted pyramid images ${}^L I_0$ and ${}^L I_1$;
end
Initialize $^{\mathbf{L}}\mathbf{u} = 0$ and $L = max\_level$;
while $L \geq 0$ do
    for $W = 0$ to max_warps do
        Re-sample coefficients of $\rho$ using ${}^L I_0$, ${}^L I_1$, and $^{\mathbf{L}}\mathbf{u}$; (Warping)
        for $Out = 0$ to max_outer_iterations do
            Solve for $^{\mathbf{L}}\mathbf{u}'$ via thresholding; (Sect. 2.4.1)
            for $In = 0$ to max_inner_iterations do
                Smoothness term iteration on $^{\mathbf{L}}\mathbf{u}'$; (Sect. 2.4.2)
            end
        end
    end
    if $L > 0$ then
        Prolongate $^{\mathbf{L}}\mathbf{u}$ to next pyramid level $L - 1$;
    end
end
```

Algorithm 2.1: Numerical scheme of the *Refinement Optical Flow* algorithm. In the numerical scheme, a super-scripted ^{L} denotes the pyramid level

the derivatives, $\nabla I = (1-\beta)\nabla I_1(\mathbf{x}+\mathbf{u}_0) + \beta \nabla I_0(\mathbf{x})$, are being used. Values for β around 0.5 show the best results in terms of optical flow accuracy. This can be explained by the fact that both images contribute to the gradient, increasing the amount of image information used.

2.4.3.5 Numerical Scheme

The general numerical implementation for a specific setting of the Refinement Optical Flow framework is as follows. Beginning with the coarsest level, (2.11) and (2.19) are iteratively solved while the solution is propagated to the next finer level.

The prior solution on each pyramid level is used to compute the coefficients of the linearized data terms on the corresponding pyramid levels. Hence, a warping step for the input images takes place every time when the solution is propagated within the pyramid and furthermore additional warps are used on each level to achieve more accurate results. For implementation on modern graphic processing units (GPUs), bi-linear warping is essentially available at no additional cost, therefore the concept of outer iterations is only used on CPU implementations.

Avoiding poor local minima is not the only advantage of the coarse-to-fine approach. It turns out that the filling-in process induced by the regularization (smoothness) occurring in texture-less region is substantially accelerated by a hierarchical scheme as well. The resulting numerical scheme is summarized in Algorithm 2.1.

The next chapter tackles the problem of illumination artifacts, which were not addressed in the Refinement Optical Flow framework. Illumination problems such as shadows, color change and reflections are a quite general problem. Using color images, a common way to tackle illumination artifacts is to change the color space representation (e.g. from RGB to HSV) and to work solely on the color itself instead of using illumination. The approach for gray value images needs to be somehow more subtle.