

An Improved Algorithm for TV- L^1 Optical Flow

A. Wedel^{1,3}, T. Pock², C. Zach⁴, H. Bischof², and D. Cremers¹

¹ Computer Vision Group, University of Bonn

² Institute for Computer Graphics and Vision, TU Graz

³ Daimler Group Research and Advanced Engineering, Sindelfingen

⁴ Department of Computer Science, University of North Carolina at Chapel Hill

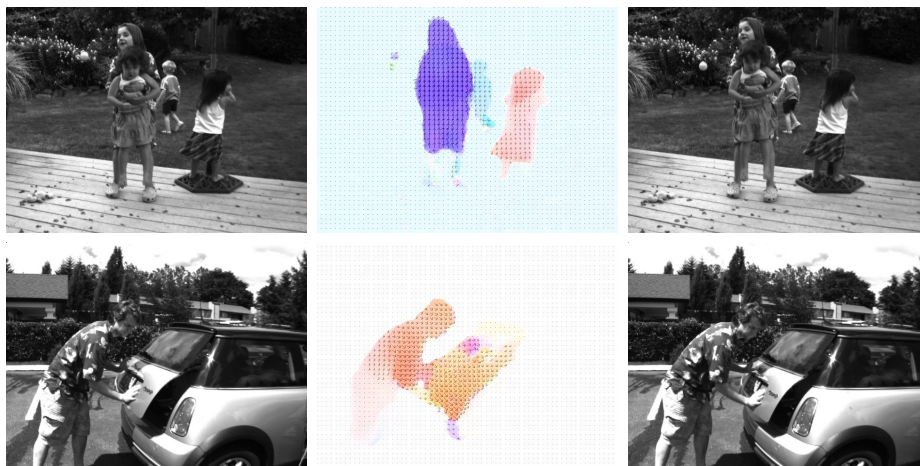


Fig. 1. Optical flow for the *backyard* and *mini cooper* scene of the Middlebury optical flow benchmark. Optical flow captures the dynamics of a scene by estimating the motion of every pixel between two frames of an image sequence. The displacement of every pixel is shown as displacement vectors on top of the commonly used flow color scheme (see Figure 5).

Abstract. A look at the Middlebury optical flow benchmark [5] reveals that nowadays variational methods yield the most accurate optical flow fields between two image frames. In this work we propose an improvement variant of the original duality based TV- L^1 optical flow algorithm in [31] and provide implementation details. This formulation can preserve discontinuities in the flow field by employing total variation (TV) regularization. Furthermore, it offers robustness against outliers by applying the robust L^1 norm in the data fidelity term.

Our contributions are as follows. First, we propose to perform a structure-texture decomposition of the input images to get rid of violations in the optical flow constraint due to illumination changes. Second, we propose to integrate a median filter into the numerical scheme to further increase the robustness to sampling artefacts in the image data. We experimentally show that very precise and robust estimation of optical flow can be achieved with a variational approach in real-time. The numerical scheme and the implementation are described in a detailed way, which enables reimplementing of this high-end method.

1 Introduction

The recovery of motion from images (see Figure 1) is a major task of biological and artificial vision systems. The objective of optical flow methods is to compute a flow field representing the motion of pixels in two consecutive image frames. Since optical flow is an highly ill-posed inverse problem, using pure intensity-based constraints results in an under-determined system of equations, which is known as the *aperture problem*. In order to solve this problem some kind of regularization is needed to obtain physically meaningful displacement fields.

In their seminal work [18], Horn and Schunck studied a variational formulation of the optical flow problem.

$$\min_{\mathbf{u}} \left\{ \int_{\Omega} |\nabla u_1|^2 + |\nabla u_2|^2 \, d\Omega + \lambda \int_{\Omega} (I_1(\mathbf{x} + \mathbf{u}(\mathbf{x})) - I_0(\mathbf{x}))^2 \, d\Omega \right\}. \quad (1)$$

Here, I_0 and I_1 is the image pair, $\mathbf{u} = (u_1(\mathbf{x}), u_2(\mathbf{x}))^T$ is the two-dimensional displacement field and λ is a free parameter. The first term (regularization term) penalizes high variations in \mathbf{u} to obtain smooth displacement fields. The second term (data term) is also known as the optical flow constraint. It assumes, that the intensity values of $I_0(\mathbf{x})$ do not change during its motion to $I_1(\mathbf{x} + \mathbf{u}(\mathbf{x}))$. The free parameter λ weighs between the data fidelity term and the regularization force. Generally speaking, \mathbf{u} registers the pixels of the source image I_0 onto the pixels of the target image I_1 .

Since the Horn-Schunck model penalizes deviations in a quadratic way, it has two major limitations. It does not allow for discontinuities in the displacement field, and it does not handle outliers in the data term robustly. To overcome these limitations, several models including robust error norms and higher order data terms have been proposed. Since discontinuities in the optical flow appear often in conjunction with high image gradients, several authors replace the homogeneous regularization in the Horn-Schunck model with an anisotropic diffusion approach [21, 29]. Others substitute the squared penalty functions in the Horn-Schunck model with more robust variants. Black and Anandan [7] apply estimators from robust statistics and obtain a robust and discontinuity preserving formulation for the optical flow energy. Aubert et al. [3] analyze energy functionals for optical flow incorporating an L^1 data fidelity term and a general class of discontinuity preserving regularization forces. Papenberg et al. [22] employ a differentiable approximation of the TV (resp. L^1) norm and formulate a nested iteration scheme to compute the displacement field.

Most approaches for optical flow computation replace the nonlinear intensity profile $I_1(\mathbf{x} + \mathbf{u}(\mathbf{x}))$ by a first order Taylor approximation to linearize the problem locally. Since such approximation is only valid for small displacements, additional techniques are required to determine the optical flow correctly for large displacements. Scale-space approaches [1] and coarse-to-fine warping (e.g. [2, 19, 9]) provide solutions to optical flow estimation with large displacements.

In several applications, such as autonomous robot navigation, it is necessary to calculate displacement fields in real-time. Real-time optical flow techniques typically consider only the data fidelity term to generate displacement fields [12, 25]. One of the first variational approaches to compute the optical flow in real-time was presented by

Bruhn et al. [10, 11]. In their work a highly efficient multi-grid approach is employed to obtain real-time or near real-time performance. The aim of their approach is very similar to our objective: obtaining robust and discontinuity preserving solutions for optical flow with highly efficient implementations. Nevertheless, we utilize a completely different solution strategy, namely a duality based TV- L^1 optical flow algorithm introduced in [31]. In the following section, we reproduce this approach before we present some improvements to increase the robustness and flow accuracy.

2 TV- L^1 Optical Flow [31]

In the basic setting two image frames I_0 and $I_1 : (\Omega \subseteq \mathbb{R}^2) \rightarrow \mathbb{R}$ are given. The objective is to find the disparity map $\mathbf{u} : \Omega \rightarrow \mathbb{R}^2$, which minimizes an image-based error criterion together with a regularization force. In this work we focus on the plain intensity difference between pixels as the image similarity score. Hence, the target disparity map \mathbf{u} is the minimizer of

$$\int_{\Omega} \left\{ \lambda \phi(I_0(\mathbf{x}) - I_1(\mathbf{x} + \mathbf{u}(\mathbf{x}))) + \psi(\mathbf{u}, \nabla \mathbf{u}, \dots) \right\} d\mathbf{x}, \quad (2)$$

where $\phi(I_0(\mathbf{x}) - I_1(\mathbf{x} + \mathbf{u}(\mathbf{x})))$ is the image data fidelity, and $\psi(\mathbf{u}, \nabla \mathbf{u}, \dots)$ depicts the regularization term. The parameter λ weighs between the data fidelity and the regularization force. Selecting $\phi(x) = x^2$ and $\psi(\nabla \mathbf{u}) = |\nabla \mathbf{u}|^2$ results in the Horn-Schunck model [18].

The choice of $\phi(x) = |x|$ and $\psi(\nabla \mathbf{u}) = |\nabla \mathbf{u}|$ yields to the following functional consisting of an L^1 data penalty term and total variation regularization:

$$E = \int_{\Omega} \left\{ \lambda |I_0(\mathbf{x}) - I_1(\mathbf{x} + \mathbf{u}(\mathbf{x}))| + |\nabla \mathbf{u}| \right\} d\mathbf{x}. \quad (3)$$

Although Eq. 3 seems to be simple, it offers computational difficulties. The main reason is that both, the regularization term and the data term, are not continuously differentiable. One approach is to replace $\phi(x) = |x|$ and $\psi(\nabla \mathbf{u}) = |\nabla \mathbf{u}|$ with differentiable approximations, $\phi_{\varepsilon}(x^2) = \sqrt{x^2 + \varepsilon^2}$ and $\psi_{\varepsilon}(\nabla \mathbf{u}) = \sqrt{|\nabla \mathbf{u}|^2 + \varepsilon^2}$, and to apply a numerical optimization technique on this slightly modified functional (e.g. [15, 9]).

In [13] Chambolle proposed an efficient and exact numerical scheme to solve the Rudin-Osher-Fatemi energy [23] for total variation based image denoising. In the following we show how this approach was adopted in [31] to the optical flow case, yielding a different approach to solve Eq. 3.

2.1 The 1D Stereo Case

In this section we restrict the disparities to be non-zero only in the horizontal direction, e.g. a normalized stereo image pair is provided. Hence, $\mathbf{u}(\mathbf{x})$ reduces to a scalar $u(\mathbf{x})$, and we use the (sloppy) notation $\mathbf{x} + u(\mathbf{x})$ for $\mathbf{x} + (u(\mathbf{x}), 0)^T$. The following derivation is based on [4], but adapted to the stereo/optical flow setting. At first, we linearize image I_1 near $\mathbf{x} + u_0$, i.e.

$$I_1(\mathbf{x} + u) = I_1(\mathbf{x} + u_0) + (u - u_0) I_1^x(\mathbf{x} + u_0),$$

where u_0 is a given disparity map and I_1^x is the derivative of the image intensity I_1 wrt. the x -direction. Using the first order Taylor approximation for I_1 means, that the following procedure needs to be embedded into an iterative warping approach to compensate for image nonlinearities. Additionally, a multi-level approach is employed to allow large disparities between the images.

For fixed u_0 and using the linear approximation for I_1 , the TV- L^1 functional (Eq. 3) now reads as:

$$E = \int_{\Omega} \left\{ \lambda |u I_1^x + I_1(\mathbf{x} + u_0) - u_0 I_1^x - I_0| + |\nabla u| \right\} d\mathbf{x}. \quad (4)$$

In the following, we denote the current residual $I_1(\mathbf{x} + u_0) + (u - u_0) I_1^x - I_0$ by $\rho(u, u_0, \mathbf{x})$ (or just $\rho(u)$ by omitting the explicit dependency on u_0 and \mathbf{x}). Moreover, we introduce an auxiliary variable v and propose to minimize the following convex approximation of Eq. 4:

$$E_{\theta} = \int_{\Omega} \left\{ |\nabla u| + \frac{1}{2\theta} (u - v)^2 + \lambda |\rho(v)| \right\} d\mathbf{x}, \quad (5)$$

where θ is a small constant, such that v is a close approximation of u . This convex minimization problem can be optimized by alternating steps updating either u or v in every iteration:

1. For v being fixed, solve

$$\min_u \int_{\Omega} \left\{ |\nabla u| + \frac{1}{2\theta} (u - v)^2 \right\} d\mathbf{x}. \quad (6)$$

This is the total variation based image denoising model of Rudin, Osher and Fatemi [23].

2. For u being fixed, solve

$$\min_v \int_{\Omega} \left\{ \frac{1}{2\theta} (u - v)^2 + \lambda |\rho(v)| \right\} d\mathbf{x}. \quad (7)$$

This minimization problem can be solved point-wise, since it does not depend on spatial derivatives of v .

An efficient solution for the first step (Eq. 6) is based on gradient descent and subsequent re-projection using the dual-ROF model [14]. It is based on a dual formulation of Eq. 6 and yields a globally convergent iterative scheme. Since this algorithm is an essential part of our method, we reproduce the relevant results from [14]:

Proposition 1 *The solution of Eq. (6) is given by*

$$u = v + \theta \mathbf{div} \mathbf{p}. \quad (8)$$

The dual variable $\mathbf{p} = [p_1, p_2]$ is defined iteratively by

$$\tilde{\mathbf{p}}^{n+1} = \mathbf{p} + \frac{\tau}{\theta} (\nabla (v + \theta \mathbf{div} \mathbf{p}^n)) \text{ and} \quad (9)$$

$$\mathbf{p}^{n+1} = \frac{\tilde{\mathbf{p}}^{n+1}}{\max \{1, |\tilde{\mathbf{p}}^{n+1}|\}} \quad (10)$$

where $\mathbf{p}^0 = \mathbf{0}$ and the time step $\tau \leq 1/4$.

Proposition 2 *The solution of the minimization task in Eq. 7 is given by the following thresholding step:*

$$v = u + \begin{cases} \lambda \theta I_1^x & \text{if } \rho(u) < -\lambda \theta (I_1^x)^2 \\ -\lambda \theta I_1^x & \text{if } \rho(u) > \lambda \theta (I_1^x)^2 \\ -\rho(u)/I_1^x & \text{if } |\rho(u)| \leq \lambda \theta (I_1^x)^2. \end{cases} \quad (11)$$

This means, that the image residual $\rho(v)$ is allowed to vanish, if the required step from u to v is sufficiently small. Otherwise, v makes a bounded step from u , such that the magnitude of the residual decreases. The proposition above can be shown directly by analyzing the three possible cases, $\rho(v) > 0$ (inducing $v = u - \lambda \theta I_1^x$), $\rho(v) < 0$ ($v = u + \lambda \theta I_1^x$) and $\rho(v) = 0$ ($v = u - \rho(u)/I_1^x$).

2.2 Generalization to Higher Dimensions

In this section we extend the method introduced in the previous section to optical flow estimation, i.e. a N -dimensional displacement map \mathbf{u} is determined from two given N -D images I_0 and I_1 . The first order image residual $\rho(\mathbf{u}, \mathbf{u}_0, \mathbf{x})$ wrt. a given disparity map \mathbf{u}_0 is now $I_1(\mathbf{x} + \mathbf{u}_0) + \langle \nabla I_1, \mathbf{u} - \mathbf{u}_0 \rangle - I_0(\mathbf{x})$. Additionally, we write u_d for the d -th component of \mathbf{u} ($d \in \{1, \dots, N\}$).

The generalization of Eq. 5 to more dimensions is the following energy:

$$E_\theta = \int_{\Omega} \left\{ \sum_d |\nabla u_d| + \sum_d \frac{1}{2\theta} (u_d - v_d)^2 + \lambda |\rho(\mathbf{v})| \right\} d\mathbf{x}. \quad (12)$$

Similar to the stereo setting, minimizing this energy can be performed by alternating optimization steps:

1. For every d and fixed v_d , solve

$$\min_{u_d} \int_{\Omega} \left\{ |\nabla u_d| + \frac{1}{2\theta} (u_d - v_d)^2 \right\} d\mathbf{x}. \quad (13)$$

This minimization problem is identical to Eq. 6 and can be solved by the same procedure. Note, that the dual variables are introduced for every dimension, e.g. Eq. 8 now reads as

$$u_d = v_d - \theta \operatorname{div} \mathbf{p}_d. \quad (14)$$

2. For \mathbf{u} being fixed, solve

$$\min_{\mathbf{v}} \sum_d \frac{1}{2\theta} (u_d - v_d)^2 + \lambda |\rho(\mathbf{v})|. \quad (15)$$

The following proposition generalizes the thresholding step from Proposition 2 to higher dimensions:

Proposition 3 *The solution of the minimization task in Eq. 15 is given by the following thresholding step:*

$$\mathbf{v} = \mathbf{u} + \begin{cases} \lambda \theta \nabla I_1 & \text{if } \rho(\mathbf{u}) < -\lambda \theta |\nabla I_1|^2 \\ -\lambda \theta \nabla I_1 & \text{if } \rho(\mathbf{u}) > \lambda \theta |\nabla I_1|^2 \\ -\rho(\mathbf{u}) \nabla I_1 / |\nabla I_1|^2 & \text{if } |\rho(\mathbf{u})| \leq \lambda \theta |\nabla I_1|^2. \end{cases} \quad (16)$$

This proposition essentially states, that the N -dimensional optimization problem can be reduced to a one-dimensional thresholding step, since \mathbf{v} always lies on the line l^\perp going through \mathbf{u} with direction ∇I_1 (for every \mathbf{x}). This can be seen as follows: The first part in Eq. 15, $\sum_d (u_d - v_d)^2 / 2\theta$, is basically the squared distance of \mathbf{v} to \mathbf{u} , and the second part, $\lambda |\rho(\mathbf{v})|$, is the unsigned distance to the line $l : \rho(\mathbf{w}) = 0$, i.e. $I_1(\mathbf{x} + \mathbf{u}_0) + \langle \nabla I_1, \mathbf{w} - \mathbf{u}_0 \rangle - I_0(\mathbf{x}) = 0$. If we consider all \mathbf{v}_μ with a fixed distance μ to \mathbf{u} , then the functional in Eq. 15 is minimized for the \mathbf{v}_μ closest to the line l (with minimal normal distance). This is also valid for the true minimizer, hence the optimum for Eq. 15 is on l^\perp . In addition, the one-dimensional thresholding step in gradient direction can be applied (Proposition 2), resulting in the presented scheme.

3 Increasing Robustness to Illumination Changes

The image data fidelity term $\phi(I_0(\mathbf{x}) - I_1(\mathbf{x} + \mathbf{u}(\mathbf{x})))$ states that the intensity values of $I_0(\mathbf{x})$ do not change during its motion to $I_1(\mathbf{x} + \mathbf{u}(\mathbf{x}))$. For many sequences this constraint is violated due to sensor noise, illumination changes, reflections, and shadows. Thus, real scenes generally show artifacts that violate the optical flow constraint. Figure 2 shows an example, where the ground truth flow is used to register two images from the Middlebury optical flow benchmark data base [5]. Although the two images are registered at the best using the ground truth flow, the intensity difference image between the source image and the registered target image reveals the violations of the optical flow constraint. Some of these regions, showing artifacts of shadow and shading reflections, are marked by blue circles in the intensity difference image.

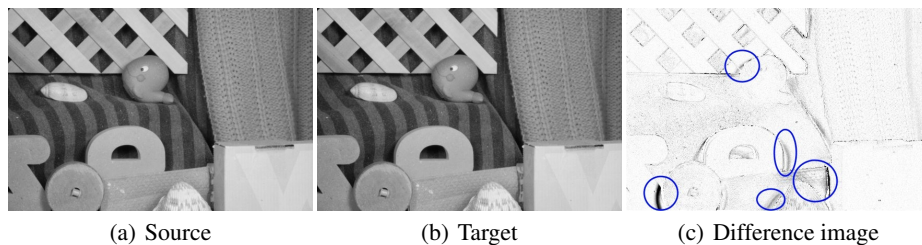


Fig. 2. The source and target images of the *rubber-whale* sequence in the Middlebury optical flow benchmark have been registered using the ground truth optical flow. Still, intensity value differences are visible due to sensor noise, reflections, and shadows. The intensity difference image is encoded from white (no intensity value difference) to black (10% intensity value difference). Pixels which are visible in a single image due to occlusion are shown in white.

A physical model of brightness changes was presented in [17], where brightness change and motion is estimated simultaneous; shading artefacts however have not been addressed. In [30] and [20] the authors used photometric invariants to cope with brightness changes, which requires color images. A common approach in literature to tackle illumination changes is to use image gradients besides, or instead of, the plain image intensity values in the data term [9]. This implies that multiple data fidelity terms have to be used and images are differentiated twice, which is known to be noisy.

Here we propose a structure-texture decomposition similar to the approach used in [26] to model the intensity value artifacts due to shading reflections and shadows. The basic idea behind this splitting technique is that an image can be regarded as a composition of a structural part, corresponding to the main large objects in the image, and a textural part, containing fine scale-details [4]. See Figure 3 for an example of such a structure-texture decomposition, also known as cartoon-texture decomposition. The expectation is, that shadows show up only in the structural part which includes the main large objects.

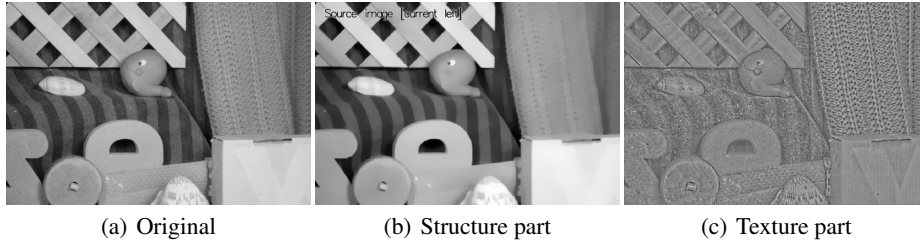


Fig. 3. The original image is decomposed into a structural part, corresponding to the main large objects in the image, and a textural part, containing fine-scale details. All images are scaled into the same intensity value range after decomposition.

The structure-texture decomposition is accomplished using the total variation based image denoising model of Rudin, Osher and Fatemi [23]. For the intensity value image $I(\mathbf{x})$, the structural part is given as the solution of

$$\min_{I_S} \int_{\Omega} \left\{ |\nabla I_S| + \frac{1}{2\theta} (I_S - I)^2 \right\} d\mathbf{x}. \quad (17)$$

The textural part $I_T(\mathbf{x})$ is then computed as the difference between the original image and its denoised version, $I_T(\mathbf{x}) = I(\mathbf{x}) - I_S(\mathbf{x})$. Figure 4 shows the intensity difference images between the source image and the registered target image using the ground truth flow for the original image and its decomposed parts. For most parts the artifacts due to shadow and shading reflections show up in the original image and the structural part. The intensity value difference using the textural part, which contains fine-scale details, is noisier than the intensity value difference in the structural part. These intensity value differences are mainly due to sensor noise and sampling artifacts while shadow and shading reflection artifacts have been almost completely removed. This is best visible in the area of the punched hole of the rotated D-shaped object.

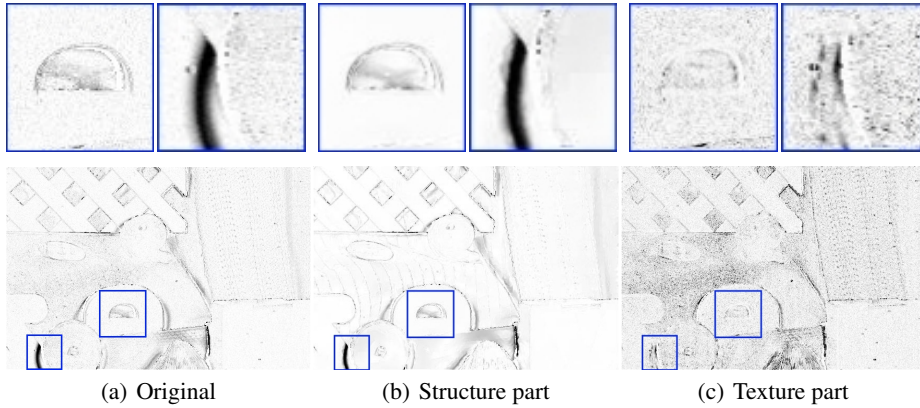


Fig. 4. Intensity difference images between the source image and the registered target image using ground truth optical flow for the original image and its structure-texture decomposed versions (intensity coding as in Figure 2). Note the presence of shading reflection and shadow artifacts in the original image and in the structure image.

This observation leads to the assumption that the computation of optical flow using the textural part of the image is not perturbed by shadow and shading reflection artifacts, which cover large image regions. To prove this assumption experimentally, we use a blended version of the textural part, $I_T(\alpha, \mathbf{x}) = I(\mathbf{x}) - \alpha I_S(\mathbf{x})$, as input for the optical flow computation. Figure 5 shows the accuracy for optical flow computation using a fixed parameter set and varying the blending factor α . The plot reveals that for larger values of α the accuracy of the optical flow is 30% better than using a small value for α . This confirms the assumption that removing large perturbations due to shadow and shading reflections yields better optical flow estimates. In the experiments we set $\alpha = 0.95$ and compute the image decomposition as follows:

The original source and target images are scaled into the range $[-1, 1]$ before computing the structure part. We use $\lambda_{ROF} = 0.125$ and 100 iterations of the re-projection step presented in Proposition 1 to solve Eq. (17). In the CPU implementation, the resulting source and target texture images are also equivalently scaled into the range $[-1, 1]$ prior to optical flow computation.

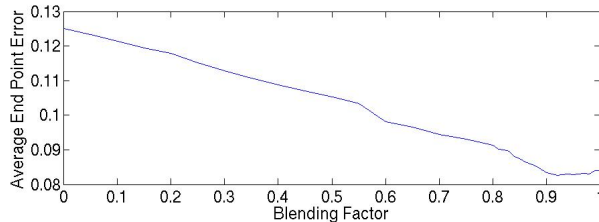


Fig. 5. The plot shows the optical flow accuracy, measured as the average end point error, using different α values for the blending of the textural part of the image (same image pair as Figure 3). The improvement using the textural part for the optical flow computation becomes visible.

4 Implementation

This section gives details on the employed numerical procedure and on the implementation for the proposed TV- L^1 optical flow approach. Although the discussion in Section 2.2 is valid for any image dimension $N \geq 2$, the discussion in this Section is specifically tailored for the case $N = 2$.

4.1 Numerical Scheme

The generally non-convex energy functional for optical flow (Eq. 3) becomes a convex minimization problem after linearization of the image intensities (Eq. 4). But this linearization is only valid for small displacements. Hence, the energy minimization procedure is embedded into a coarse-to-fine approach to avoid convergence to unfavorable local minima. We employ image pyramids with a down-sampling factor of 2 for this purpose. The resulting numerical scheme is summarized in Algorithm 1.

```

Input: Two intensity images  $I_0$  and  $I_1$ 
Output: Flow field  $\mathbf{u}$  from  $I_0$  to  $I_1$ 
Preprocess the input images; (Sec. 3)
for  $L = 0$  to  $max\_level$  do
    Calculate restricted pyramid images  ${}^L I_0$  and  ${}^L I_1$ ;
end
Initialize  ${}^L \mathbf{u} = 0$ ,  ${}^L \mathbf{p} = 0$ , 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  ${}^L \mathbf{u}$ ; (Warping)
        for  $Out = 0$  to  $max\_outer\_iterations$  do
            Solve for  ${}^L \mathbf{v}$  via thresholding; (Eq. 16)
            for  $In = 0$  to  $max\_inner\_iterations$  do
                Perform one iteration step to solve for  ${}^L \mathbf{u}$ ; (Prop. 1)
            end
            Median filter  ${}^L \mathbf{u}$ ;
        end
    end
    if  $L > 0$  then
        Prolongate  ${}^L \mathbf{u}$  and  ${}^L \mathbf{p}$  to next pyramid level  $L - 1$ ;
    end
end

```

Algorithm 1: Numerical scheme of the TV- L^1 optical flow. In the numerical scheme, a super-scripted L denotes the pyramid level.

Beginning with the coarsest level, we solve Eq. 3 at each level of the pyramid and propagate the solution to the next finer level. This solution is further used to compute the coefficients of the linear residual function ρ by sampling I_0 and I_1 using the corresponding pyramid levels. Thus, the warping step for I_1 takes place every time, the solution is propagated across pyramid levels. We use additional warps on each level to get more accurate results. 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 occurring in texture-less region is substantially accelerated by a hierarchical scheme as well. In the following subsections the single steps of the numerical scheme are outlined and implementation details are provided.

4.2 Pyramid Restriction and Prolongation

The pyramid restriction and prolongation operations for image intensities, flow vectors, and the dual variable \mathbf{p} are quite different. While gray values can simply be averaged, flow vectors need to be scaled with the scaling factor between the pyramid levels to yield valid displacement vectors on every pyramid level. In our case we employ image pyramids with a down-sampling factor of 2.

The restriction operator, which is used for the intensity images is a combination of a low pass 5×5 binomial filter and subsequent down-sampling [24]. That is, odd rows and columns are removed from the image (note, that such procedure does require the size of the input image to be a power of 2 times the size of the lowest resolved image). The mask used is

$$\frac{1}{16} \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \times \frac{1}{16} [1 \ 4 \ 6 \ 4 \ 1] = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}. \quad (18)$$

The prolongation operator up-samples the image, that is, inserts odd zero rows and columns, and then applies the 5×5 binomial filter multiplied by 4 to it. Here we have to differentiate between up-sampling of flow vectors \mathbf{u} , which have to be multiplied by a factor of 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.

4.3 Outer Iteration: Re-sampling the Data Term Coefficients via Warping

Similarly to the 1D stereo case (Sec. 2.1), the image I_1 is linearized using the first order Taylor approximation near $\mathbf{x} + \mathbf{u}_0$, where \mathbf{u}_0 is a given optical flow map:

$$I_1(\mathbf{x} + \mathbf{u}) = I_1(\mathbf{x} + \mathbf{u}_0) + (\mathbf{u} + \mathbf{u}_0) \nabla I_1(\mathbf{x} + \mathbf{u}_0). \quad (19)$$

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

$$\rho(\mathbf{u}) = \mathbf{u} \nabla I_1(\mathbf{x} + \mathbf{u}_0) + \underbrace{I_1(\mathbf{x} + \mathbf{u}_0) - \mathbf{u}_0 \nabla I_1(\mathbf{x} + \mathbf{u}_0) - I_0(\mathbf{x})}_c, \quad (20)$$

where the right part, denoted by c , is independent of \mathbf{u} , and hence fixed. We use bi-cubic look-up to calculate the intensity value $I_1(\mathbf{x} + \mathbf{u}_0)$ and the derivatives of I_1 (bi-linear lookup on the GPU). The derivatives on the input images are approximated using the five-point stencil $\frac{1}{12} \begin{bmatrix} -1 & 8 & 0 & -8 & 1 \end{bmatrix}$. If the bi-cubic look-up falls onto or outside the original image boundary, a value of 0 is returned.

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 better results are achieved. Figure 6 shows the accuracy for blended versions of the derivative $\nabla I = (1 - \beta)\nabla I_1(\mathbf{x} + \mathbf{u}_0) + \beta\nabla I_0(\mathbf{x})$ keeping all other parameters fix. 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 redundancy. In our experiments we use a fixed value of $\beta = 0.4$. A similar approach has been proposed for symmetric KLT tracking by Birchfield in [6].

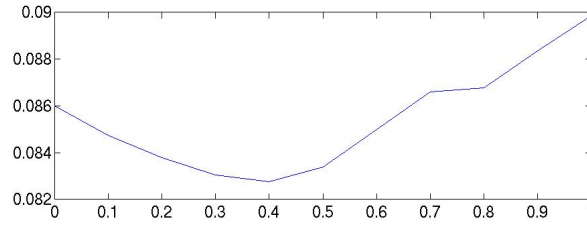


Fig. 6. The plot shows the optical flow accuracy, measured as the average end point error, using different β values for the blending of the gradients from image I_0 and I_1 (same image pair as Figure 3). The improvement using a blended version of the gradients for the optical flow computation becomes visible.

4.4 Inner Iteration: Minimization Procedure of \mathbf{u} and \mathbf{v}

Within every outer iteration (Proposition 3 followed by Proposition 1), a given number of fixed-point scheme steps (inner iterations) are performed to update all \mathbf{p}_d (and therefore \mathbf{u} , Proposition 1), followed by a median filtering of \mathbf{u} .

The implementation of Proposition 1 uses backward differences to approximate $\text{div } \mathbf{p}$ and forward differences for the numerical gradient computation in order to have mutually adjoint operators [13].

The discrete version of the forward difference gradient $(\nabla u)_{i,j} = ((\nabla u)_{i,j}^1, (\nabla u)_{i,j}^2)$ at pixel position (i, j) for a data field of width N and height M is defined as

$$(\nabla u)_{i,j}^1 = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } i < N \\ 0 & \text{if } i = N \end{cases} \quad (21)$$

and

$$(\nabla u)_{i,j}^2 = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } j < M \\ 0 & \text{if } j = M \end{cases}. \quad (22)$$

The discrete version of the backward differences divergence operator is

$$(\mathbf{div} \mathbf{p})_{i,j} = \begin{cases} p_{i,j}^1 - p_{i-1,j}^1 & \text{if } 1 < i < N \\ p_{i,j}^1 & \text{if } i = 1 \\ -p_{i-1,j}^1 & \text{if } i = N \end{cases} + \begin{cases} p_{i,j}^2 - p_{i,j-1}^2 & \text{if } 1 < j < M \\ p_{i,j}^2 & \text{if } j = 1 \\ -p_{i,j-1}^2 & \text{if } j = M \end{cases}. \quad (23)$$

The iterative re-projection scheme to update \mathbf{u} using the a quadratic coupling term with \mathbf{v} essentially assumes the differences between \mathbf{v} and \mathbf{u} to be Gaussian. After updating \mathbf{u} , we still find that the solution contains outliers. With the median filtering of \mathbf{u} we discard these outliers successfully. The median filter employed is a 3×3 median filter, which can be efficiently implemented [16].

4.5 Acceleration by Graphics Processing Units

Numerical methods working on regular grids, e.g. rectangular image domains, can be effectively accelerated by modern graphics processing units (GPUs). We employ the huge computational power and the parallel processing capabilities of GPUs to obtain a fully accelerated implementation of our optical flow approach. The GPU-based procedure is essentially a straightforward CUDA implementation of the numerical scheme in Algorithm 1. We currently use a fixed but tunable number of warps and iterations on each level in our implementations. Results using both, the CPU version and the GPU-based optical flow can be found in the next section.

5 Results

In this section we provide three sets of results. The first set quantitatively evaluates the accuracy increase for the proposed improved optical flow algorithm on the Middlebury flow benchmark. The benchmark provides a training data set where the ground truth optical flow is known and an evaluation set used for a comparison against other algorithms in literature. For visualization of the flow vectors we used the color coding scheme proposed in [5] (See also Figure 5).

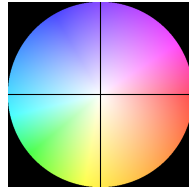
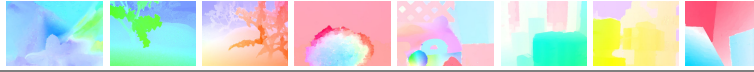


Fig. 7. Color coding of the flow vectors: Direction is coded by hue, length is coded by saturation.

The second set evaluates real scenes, taken from a moving vehicle. It demonstrates the performance of the improved optical flow algorithm under different illumination conditions and under large image motion.

In the third set of results we show optical flow results of our core real-time implementation (no texture images and no median filter) on a graphics card to evaluate our algorithm in indoor scenes.



	Dimetrodon	Grove2	Grove3	Hydrangea	RubberWhale	Urban2	Urban3	Venus	
Performance	P(GPU)	0.259	0.189	0.757	0.258	0.218	0.652	1.069	0.482
	P	0.236	0.190	0.803	0.240	0.302	0.598	0.897	0.486
	P-MF(GPU)	0.224	0.173	0.671	0.251	0.183	0.508	0.889	0.433
	P-MF	0.202	0.161	0.666	0.236	0.161	0.468	0.679	0.428
	P- I_T -MF(GPU)	0.186	0.200	0.743	0.186	0.118	0.487	1.026	0.314
	P- I_T -MF	0.171	0.191	0.730	0.173	0.109	0.390	0.812	0.311
	TV- L^1 -improved	0.190	0.154	0.665	0.147	0.092	0.319	0.630	0.260

Table 1. Evaluation results on the Middlebury training data. The evaluation is splitted into real-time *Performance* results and the results of the proposed TV- L^1 -improved algorithm, employing additional warps and bi-cubic lookup. The table shows the average end point error of the estimated flow fields. Parameters have been carefully chosen for algorithm comparison (see text for parameters and run-time).

	Algorithm	Processor	Avg. Accuracy	Run-time
Performance	P(GPU)	NVidia® GeForce® GTX 285	0.486	0.039 [sec]
	P	Intel® Core™2 Extreme 3.0GHz	0.468	0.720 [sec]
	P-MF(GPU)	NVidia® GeForce® GTX 285	0.416	0.055 [sec]
	P-MF	Intel® Core™2 Extreme 3.0GHz	0.375	0.915 [sec]
	P- I_T -MF(GPU)	NVidia® GeForce® GTX 285	0.408	0.061 [sec]
	P- I_T -MF	Intel® Core™2 Extreme 3.0GHz	0.361	1.288 [sec]

Table 2. Run-time comparison for the **Performance** section in Table 1. Using the parallel power of a GPU yields performance gain at the cost of accuracy loss. The run-time is measured on the *Grove3* test image (640×480 px).

5.1 Evaluation on the Middlebury Benchmark

The performance section in Table 1 compares real-time capable implementations for optical flow. Both, the TV- L^1 optical flow algorithm and the image decomposition described in Section 3, employ the Rudin-Osher-Fatemi denoising algorithm. This denoising step can be efficiently implemented on modern graphics cards, putting up with small accuracy losses: For parallel processing, the iterative denoising (Proposition 1) is executed on sub-blocks of the image in parallel, where boundary artifacts may occur. Hence, high accuracy is exchanged versus run-time performance (see Table 2). The measured timings do *not* include the image uploads to video memory and the final visualization of the obtained displacement field. These times are included in the timings of the optical flow algorithm for video sequences in Section 5.3.

In all three algorithm settings, P, P-MF, and P- I_T -MF, the linearized optical flow constraints (19) is used as data term. The number of outer iterations is set to one. In the plain version, algorithm P, 5 inner iterations are used in every warping step. The number of refinement warps on every pyramid level was set to 25. The parameter settings are $\lambda = 25$ and $\theta = 0.2$. Gray value look-up is bi-linear, as this can be done without additional costs on modern graphics cards. The image gradient is computed via central derivatives from the average of both input images.

The P-MF algorithm extends the basic algorithm by an additional Median filter step, hence 5 iterations of the Proposition 1, followed by a median filter step, are performed for each warp. The Median filter makes the whole scheme more robust against outliers. For this reason the influence of the data term, weighted by λ can be increased to $\lambda = 50$. All other parameters are kept fix.

In the third algorithm, P- I_T -MF, the textural part of the image is used, as described in Section 3. Note that *for real-time purposes* the input images are only scaled into the range $[-1, 1]$ once, prior to texture extraction, by using the maximum gray value. For real-time computation the min/max computation in the texture image is quite time-consuming on a GPU. Again the increase of accuracy at the cost of a longer execution time can be seen in the quantitative evaluation. It is interesting to note that only the flow fields for the *real scenes* within the test set benefit from the image decomposition. The optical flow for the *rendered scenes*, Grove and Urban, is actually worse. This is not surprising as texture-extraction removes some structure information in the images; such procedure is only beneficial if the images contain illumination artifacts. Because this is the fact for all natural scenes (which are for obvious reasons more interesting and challenging), in the remaining experiments the texture-structure decomposition is performed inherently.

In the settings for the proposed TV- $L1$ -*improved* algorithm we set the focus on accuracy. For this, we use 35 warps, 5 outer iterations, and 1 inner iteration. We set $\lambda = 30$ and $\theta = 0.25$, and use bi-cubic lookup as well as five-point differences for the gradients. The result on the test data set are shown in the last row of Table 1. Run-time on the *Grove3* sequence was 3.46 seconds. Figure 8 shows the benchmark results. Currently, as on October 22nd 2008, there are results of 19 different optical flow algorithms in the benchmark. Our method outperforms all approaches in terms of angle error and end point error. Figures 14 and 15 show the obtained results for all eight evaluation sequences. For most part, the remaining flow errors are due to occlusion artifacts.

Average angle error	Army (Hidden texture)			Mepion (Hidden texture)			Schefflera (Hidden texture)			Wooden (Hidden texture)			Grove (Synthetic)			Urban (Synthetic)			Yosemite (Synthetic)			Teddy (Stereo)			
	avg.	GT	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	
	rank	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest
TV-L1-improved [19]	3.8	3.36	9.63	2.62	2.82	10.7	2.23	6.50	15.8	2.73	3.80	21.3	1.76	3.34	4.38	2.39	5.97	18.1	5.67	3.57	4.92	3.43	4.01	9.84	3.44
F-TV-L1 [18]	5.2	5.44	12.5	5.69	5.48	15.0	4.03	7.48	16.3	3.42	5.08	23.3	2.81	3.42	4.34	3.03	4.05	15.1	3.18	2.43	3.92	1.87	3.90	9.35	2.61
Brow et al. [7]	6.2	4.80	14.4	4.29	4.05	13.5	3.71	6.63	16.0	7.26	5.22	22.7	3.22	4.56	6.09	3.40	3.87	17.9	3.41	2.02	3.76	1.18	5.14	11.9	4.28
Fusion [9]	6.5	4.43	13.7	4.08	2.47	8.91	2.24	3.70	9.68	3.12	3.68	19.8	2.54	4.26	5.16	4.31	6.32	16.8	6.15	4.55	5.78	3.10	7.12	13.6	11.7
Dynamic MRF [10]	7.1	4.58	12.4	4.14	3.25	13.9	2.27	6.02	16.8	2.36	4.39	22.6	2.51	3.61	4.55	3.46	6.81	22.2	6.78	2.41	3.48	3.69	2.26	16.7	10.2
SegOF [13]	7.2	5.85	13.5	3.98	7.40	14.9	8.13	8.55	17.3	9.01	8.50	19	18.1	3.90	4.53	4.81	6.57	21.7	6.81	1.65	3.49	1.08	3.71	9.23	3.63
CBF [15]	7.3	3.95	10.1	3.44	3.70	10.6	3.85	5.64	13.5	3.34	3.71	21.5	1.99	4.36	5.50	3.55	11.3	19.1	9.05	6.79	7.37	11.6	5.50	11.8	5.66
GraphCuts [17]	8.4	6.25	14.3	5.53	8.80	20.1	8.61	17.91	15.4	10.9	4.88	19.0	3.05	3.78	4.71	3.94	8.74	18.4	6.39	4.04	4.87	4.85	6.35	12.2	6.05

(a) Average angle error on the Middlebury optical flow benchmark.

Average end point error	Army (Hidden texture)			Mepion (Hidden texture)			Schefflera (Hidden texture)			Wooden (Hidden texture)			Grove (Synthetic)			Urban (Synthetic)			Yosemite (Synthetic)			Teddy (Stereo)			
	avg.	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1	GT	im0	im1
	rank	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest	all	disc	untest
TV-L1-improved [19]	3.7	0.09	0.26	0.07	0.20	0.71	0.16	0.53	1.18	0.22	0.21	1.24	0.11	0.90	1.31	0.72	1.51	1.93	0.84	0.18	0.17	0.31	0.73	1.62	0.87
Fusion [9]	5.1	0.11	0.34	0.10	0.19	0.69	0.16	0.29	0.66	0.23	0.20	1.19	0.14	1.07	1.42	1.22	1.35	1.49	0.89	0.20	0.20	0.26	1.02	2.07	1.39
F-TV-L1 [18]	5.3	0.14	0.35	0.14	0.34	0.98	0.26	0.59	1.19	0.26	0.27	1.36	0.16	0.90	1.30	0.76	0.54	1.82	0.36	0.13	0.15	0.20	0.68	1.58	0.66
Brow et al. [7]	6.0	0.12	0.37	0.11	0.31	0.97	0.28	0.49	1.11	0.49	0.28	1.28	0.19	1.13	1.57	1.11	1.02	2.02	0.69	0.18	0.13	0.11	0.93	2.00	1.07
Dynamic MRF [10]	6.5	0.12	0.34	0.11	0.22	0.89	0.16	0.44	1.13	0.20	0.24	1.29	0.14	1.11	1.52	1.13	1.54	2.37	0.93	0.13	0.12	0.31	1.27	2.33	1.66
CBF [15]	6.6	0.10	0.28	0.09	0.23	0.79	0.29	0.45	0.98	0.24	0.21	1.22	0.13	0.96	1.39	0.74	2.32	2.19	1.29	0.34	0.27	0.85	0.86	1.78	1.06
SegOF [13]	7.2	0.15	0.36	0.10	0.57	1.16	0.59	0.68	1.1	1.24	0.64	0.32	1.18	1.18	1.50	1.47	1.83	2.09	0.96	0.08	0.13	0.12	0.70	1.50	0.69
Second-order prior [11]	7.6	0.10	0.30	0.08	0.22	0.85	0.15	0.57	1.28	0.23	0.20	1.14	0.11	1.13	1.55	1.03	2.52	2.45	1.25	0.82	0.35	1.09	0.95	1.92	1.07

(b) Average end point error on Middlebury optical flow benchmark.

Fig. 8. Error measurements on the Middlebury optical flow benchmark as on October 22nd 2008. The proposed method (*TV-L1-improved*) outperforms other current state-of-the-art methods for optical flow on the Middlebury optical flow benchmark in both measurement categories, angle error and end point error.

5.2 Traffic Scenes

The computation of optical flow is important to understand the dynamics of a scene. We evaluated our optical flow in different scenarios under different illumination conditions (night, day, shadow). Images are taken from a moving vehicle where the camera monitors the road course ahead.

The first experiment in Figure 9 shows the optical flow computation on an image sequence with a person running from the right into the driving corridor. Due to illumination changes in the image (compare the sky region for example) and severe vignetting artifacts (images intensity decreases circular from the image middle), standard optical flow computation fails. Using the proposed structure-texture decomposition, a valid flow estimation is still possible. Note the reflection (note: this is not a *shading reflection*) of the moving person on the engine hood which is only visible in the structure-texture decomposed images. Artifacts due to vignetting and illumination change are not visible in the structure-texture decomposed images. This demonstrates the increase in robustness for the optical flow computation under illumination changes using the proposed decomposition of the input images.

A second example in Figure 10 shows a scene at night with reflections on the ground plane. In the intensity images the scene is very dark and not much structure is visible. The structure-texture decomposed images reveal much more about the scene. Note, that this information is also included in the intensity image but most structure in the original images is visible in the cloud region. The figure shows the optical flow using the decomposed images. Note the correct flow estimation of the street light on the left side.

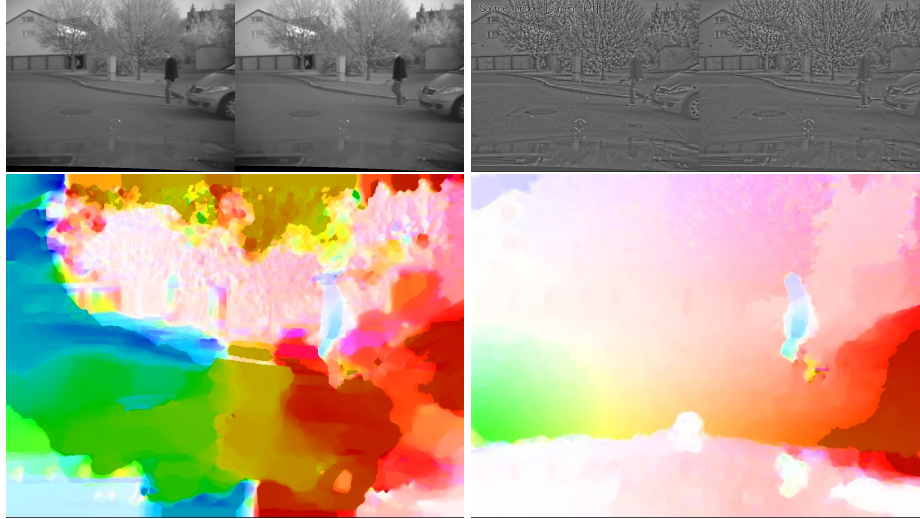


Fig. 9. Optical flow computation with illumination changes. Due to illumination changes, the optical flow constraint in the two input images (*upper left images*) is violated and flow computation on the pixel intensities (*left*) fails. Using the structure-texture decomposed images (*upper right images*), a valid flow estimation is still possible (*right side*). The optical flow color is saturated for flow vector length above $15px$.

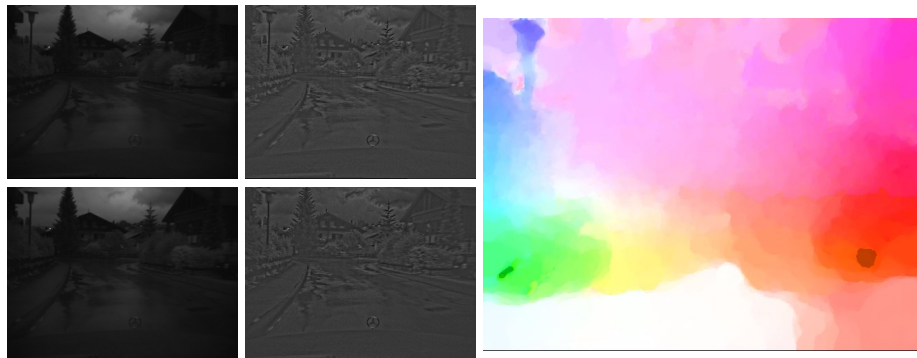


Fig. 10. Computation of optical flow for a night scene. The left images are the original intensity images. The middle images are the structure-texture decomposed images used for optical flow computation. The optical flow result is shown on the right, where flow vectors with length above $10px$ are saturated in the color coding.

The next two examples demonstrate the accurate optical flow computation for large displacements. In Figure 11 the image is taken while driving under a bridge on a country road. Note, that the shadow edge of the bridge is visible in the original images but not in the decomposed image. The large flow vectors on the reflector post are correctly matched. Only in the vicinity of the car optical flow is perturbed due to missing texture on the road surface.

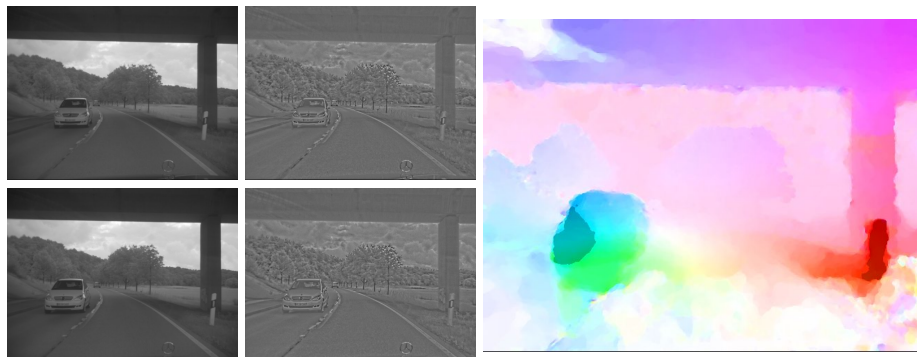


Fig. 11. The scene shows the computation of optical flow with large displacement vectors. The original input images are shown on the *left*. The *middle* images are the blended structure-texture images. Flow vectors above 20px are color-saturated in the optical flow color image.

Figure 12 shows a scene with shadows on the road. The structure-texture decomposed image reveals the structure on the road surface better than the original intensity images. We have used different scales for the optical flow color scheme to demonstrate the accuracy of our optical flow algorithm. Although nothing about epipolar geometry is used in the flow algorithm (as opposed to e. g. [27]), the effect of expansion (and hence depth) corresponding to flow length becomes visible. Note, that optical flow for the reflection posts is correctly estimated even for flow length above 8px. Optical flow is correctly estimated for the road surface up to 30px. The shadows in the scene have no negative impact on the flow calculation. The engine hood behaves like a mirror and optical flow on the engine hood is perturbed due to reflections. Although the optical flow for the engine hood is very much different for flow vectors on the road surface, this has no negative impact on the estimation of the optical flow for the road surface. Note the accurate flow discontinuity boundary along the engine hood.

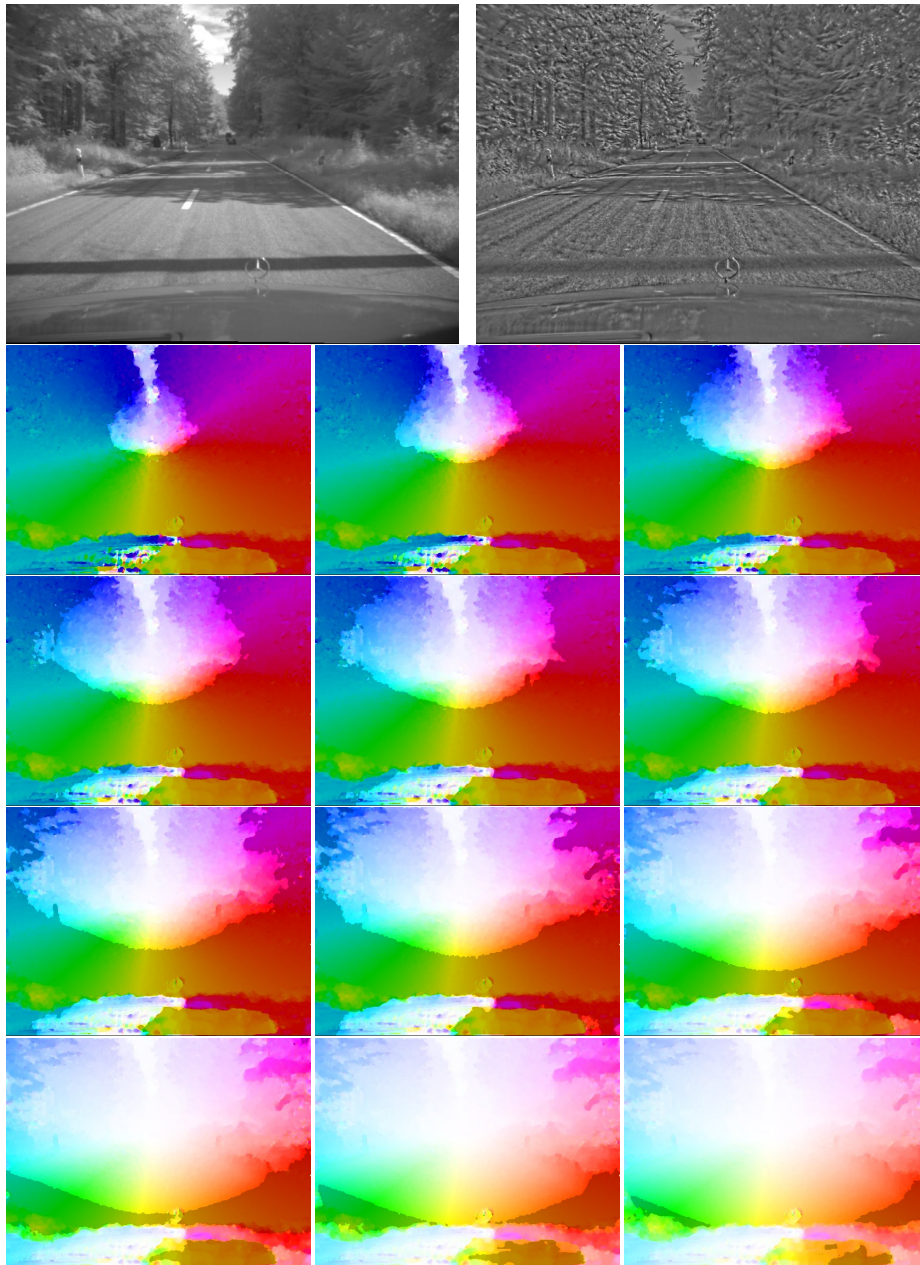


Fig. 12. Optical flow field for the scene depicted in the upper left with the original and structure-texture image. The flow is saturated for flow vector length above 1, 2, 3, 4, 5, 6, 8, 10, 15, 20, 25, 30 pixels from left to right.

5.3 Real-Time Optical Flow

We provide timing results for our optical flow approach depicted in Table 3. We used a standard personal computer equipped with a 2.13 GHz Core2-Duo CPU, 2 GB of main memory and a NVidia GTX280 graphics card. The computer runs a 64-bit Linux operating system and we used a recent NVidia graphics driver. The timings in Table 3 are given in frames per second for the depicted fixed number of outer iterations on each level of the image pyramid. We used one warp on each level, the number of fixed point steps was set to 5. The measured timings include the image uploads to video memory and the final visualization of the obtained displacement field. The timing results indicate, that real-time performance of 32 frames per second can be achieved at a resolution of 512×512 pixels. Frames from a live video demo application are shown in Figure 13, which continuously reads images from a firewire camera and visualizes the optical flow for consecutive frames. Note that the entire algorithm (including the building of the image pyramids) is executed on the GPU. The only part of the host computer is to upload the images on the GPU. In [11] Bruhn et al. obtained a performance of about 12 frames per second for 160×120 images and a variational model very similar to our TV- L^1 model. From this we see that our approach is about 26 times faster. However we should also note that their approach is computed on the CPU.

Graphics Card: NVidia GTX280			
Image resolution	25 Iterations	50 Iterations	100 Iterations
128×128	153	81	42
256×256	82	44	23
512×512	32	17	9

Table 3. Observed frame rates at different image resolutions and with varying number of outer iterations on our tested hardware.

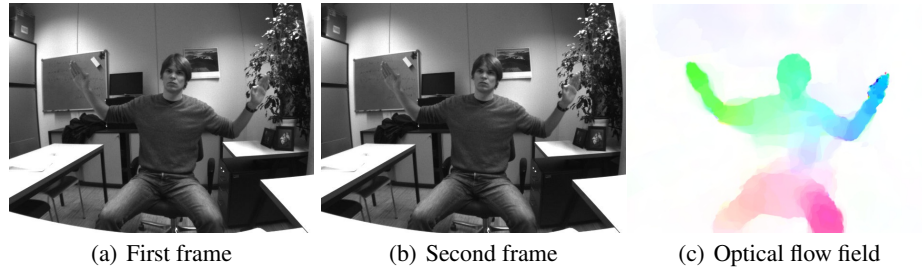


Fig. 13. Captured frames and generated optical flow field using our live video application. The image resolution is 640×480 . The performance is about 30 frames per second in this setting.

6 Conclusion and Future Research

We have proposed an improved algorithm for the TV- L^1 optical flow method of [31]. We improved the core algorithm using blended versions of the image gradients and a median filter to reject flow outliers. In this paper, the numerical scheme was outlined which can efficiently be implemented on either CPUs or modern graphics processing units. We gave implementation details and showed that the proposed improvements increase the accuracy of the optical flow estimation.

We additionally proposed to use a blended version of the structure-texture decomposition, originally proposed for optical flow computation in [26]. The decomposition of the input image into its structural and textural parts allows to minimize illumination artifacts due to shadows and shading reflections. We showed that this leads to more accurate results in the optical flow computation.

Our improved algorithm for TV- L^1 optical flow was evaluated on the Middlebury optical flow benchmark, showing state-of-the-art performance. Our proposed algorithm is solely based on the image intensities in terms of gray values. Future work includes the extension of our approach to handle color images as well.

An interesting research area is the extension of the proposed optical flow algorithms to use multiple data terms. One direct application is the computation of scene flow, incorporating three data terms [28]. We are currently investigating extensions of the proposed optical flow method to adopt it to this stereo scene flow case.

The edge preserving nature of total variation can be enhanced, if a suitable weighted TV-norm/active contour model is applied [8]. Future work will address the incorporation of such feature for stereo and optical flow estimation.

References

1. L. Alvarez, J. Weickert, and J. Sánchez. A scale-space approach to nonlocal optical flow calculations. In *Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pages 235–246, 1999.
2. P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Int. J. Comput. Vision*, 2:283–310, 1989.
3. G. Aubert, R. Deriche, and P. Kornprobst. Computing optical flow via variational techniques. *SIAM J. Appl. Math.*, 60(1):156–182, 1999.
4. J.-F. Aujol, G. Gilboa, T. Chan, and S. Osher. Structure-texture image decomposition—modeling, algorithms, and parameter selection. *Int. J. Comput. Vision*, 67(1):111–136, 2006.
5. S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black, and R. Szeliski. A Database and Evaluation Methodology for Optical Flow. In *ICCV93*, pages 1–8, 2007.
6. S. Birchfield. Derivation of Kanade-Lucas-Tomasi Tracking Equation. *Technical Report*, 1997.
7. M.J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *ICCV93*, pages 231–236, 1993.
8. X. Bresson, S. Esedoglu, P. Vandergheynst, J. Thiran, and S. Osher. Fast Global Minimization of the Active Contour/Snake Model. *Journal of Mathematical Imaging and Vision*, 2007.
9. T. Brox, A. Bruhn, N. Papenbergh, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, pages 25–36, 2004.

10. A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Variational optical flow computation in real time. *IEEE Transactions on Image Processing*, 14(5):608–615, 2005.
11. A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *Int. J. Comput. Vision*, 70(3):257–277, 2006.
12. T. A. Camus. Real-time quantized optical flow. *Journal of Real-Time Imaging*, 3:71–86, 1997. Special Issue on Real-Time Motion Analysis.
13. A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1–2):89–97, 2004.
14. A. Chambolle. Total variation minimization and a class of binary MRF models. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 136–152, 2005.
15. T. F. Chan, G. H. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. In *ICAOS '96 (Paris, 1996)*, volume 219, pages 241–252, 1996.
16. N. Devillard. Fast median search: an ANSI C implementation. em <http://www.eso.org/ndevilla/median/>, 1998, visited Oct. 2008.
17. H. Haussecker and DJ Fleet. Estimating Optical Flow with Physical Models of Brightness Variation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):661–674, 2001.
18. B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
19. E. Mémin and P. Pérez. Hierarchical estimation and segmentation of dense motion fields. *Int. J. Comput. Vision*, 46(2):129–155, 2002.
20. Y. Mileva, A. Bruhn, and J. Weickert. Illumination-robust Variational Optical Flow with Photometric Invariants. *Pattern Recognition (Proc. DAGM)*, Heidelberg, Germany, 152–159, 2007.
21. H.-H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 8:565–593, 1986.
22. N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *Int. J. Comput. Vision*, pages 141–158, 2006.
23. L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.
24. E. Stewart. Intel Integrated Performance Primitives: How to Optimize Software Applications Using Intel IPP, Intel Press, 2004, isbn 0971786135.
25. R. Strzodka and C. Garbe. Real-time motion estimation and visualization on graphics cards. In *IEEE Visualization 2004*, pages 545–552, 2004.
26. W. Trobin, T. Pock, D. Cremers, and H. Bischof. An Unbiased Second-Order Prior for High-Accuracy Motion Estimation. *Pattern Recognition (Proc. DAGM)*, June 2008.
27. A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality TV- L^1 flow with fundamental matrix prior. *Image and Vision Computing New Zealand*, November 2008.
28. A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers. Efficient Dense Scene Flow from Sparse or Dense Stereo Data. In *European Conference on Computer Vision (ECCV)*, pages 739–751, October 2008.
29. J. Weickert and T. Brox. Diffusion and regularization of vector- and matrix-valued images. *Inverse Problems, Image Analysis and Medical Imaging. Contemporary Mathematics*, 313:251–268, 2002.
30. J. van de Weijer and T. Gevers. Robust Optical Flow from Photometric Invariants. *International Conference on Image Processing*, 2004.
31. C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV- L^1 Optical Flow. *Pattern Recognition (Proc. DAGM)*, Heidelberg, Germany, 214–223, 2007.

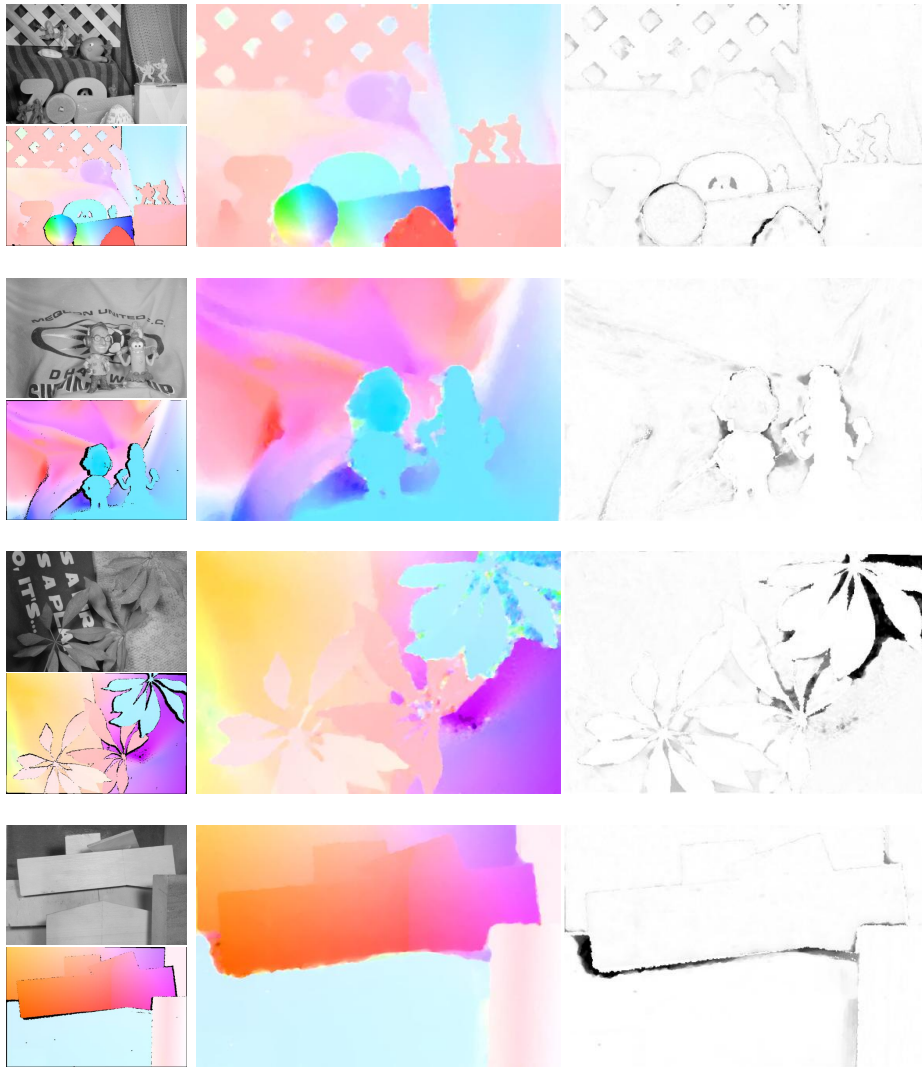


Fig. 14. Optical flow results for the *army*, *mequon*, *schefflera*, and *wooden* sequence of the Middlebury flow benchmark. The left images show the first input image and the ground truth flow. The middle image shows the optical flow using the proposed algorithm. The right image shows the end point error of the flow vector, where black corresponds to large errors.

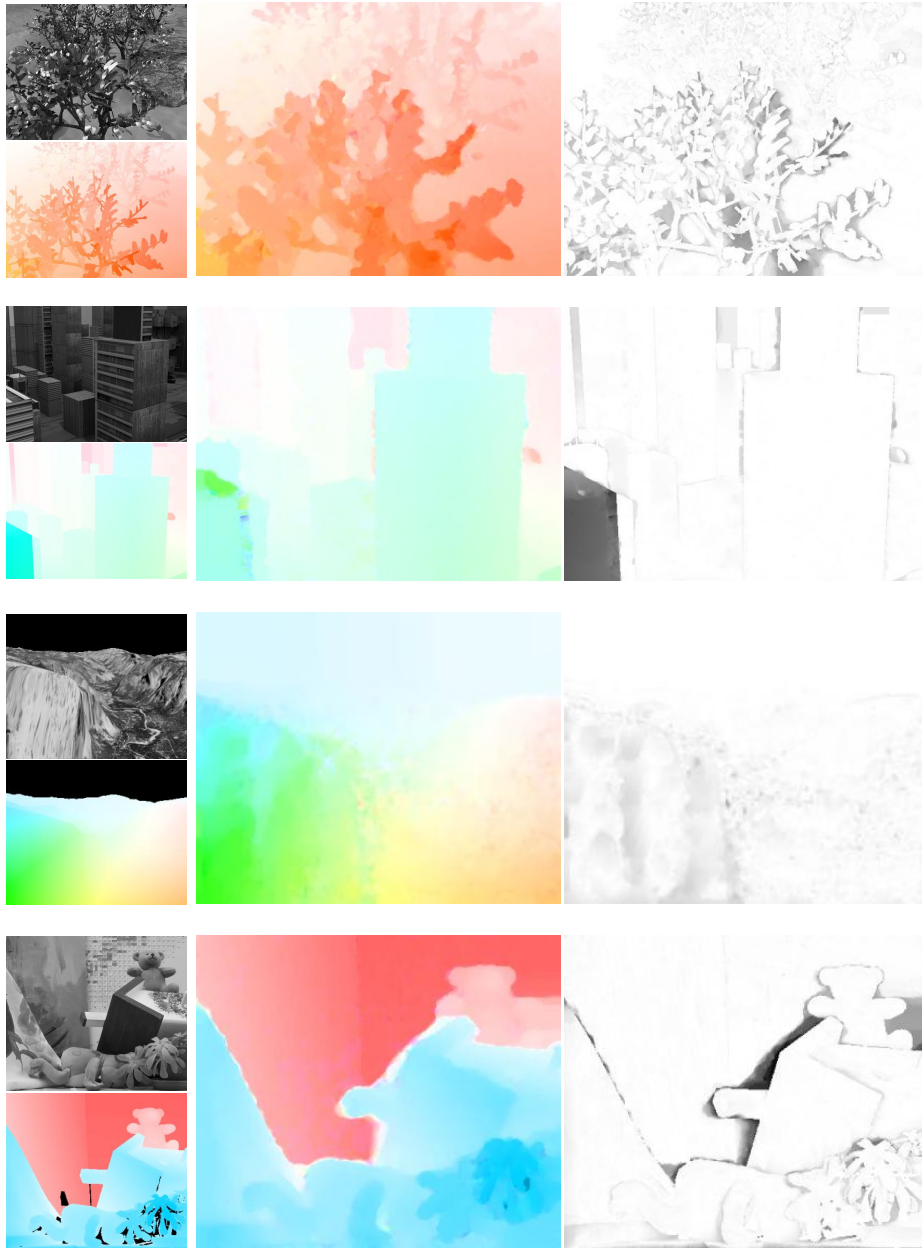


Fig. 15. (continued) Optical flow results for the *grove*, *urban*, *yosemite*, and *teddy* sequence of the Middlebury flow benchmark. The left images show the first input image and the ground truth flow. The middle image shows the optical flow using the proposed algorithm. The right image shows the end point error of the flow vector, where black corresponds to large errors.