TV-L^1 Optical Flow for Vector Valued Images

Lars Lau Rakêt[1], Lars Roholm[2], Mads Nielsen[1], and François Lauze[1]

[1] Department of Computer Science, University of Copenhagen, Denmark
{larslau, madsn, francois}@diku.dk
[2] IT University of Copenhagen, Denmark
laro@itu.dk

Abstract. The variational TV-L^1 framework has become one of the most popular and successful approaches for calculating optical flow. One reason for the popularity is the very appealing properties of the two terms in the energy formulation of the problem, the robust L^1-norm of the data fidelity term combined with the total variation (TV) regularization that smoothes the flow, but preserve strong discontinuities such as edges. Specifically the approach of Zach et al. [1] has provided a very clean and efficient algorithm for calculating TV-L^1 optical flows between grayscale images. In this paper we propose a generalized algorithm that works on vector valued images, by means of a generalized projection step. We give examples of calculations of flows for a number of multi-dimensional constancy assumptions, e.g. gradient and RGB, and show how the developed methodology expands to any kind of vector valued images. The resulting algorithms have the same degree of parallelism as the case of one-dimensional images, and we have produced an efficient GPU implementation, that can take vector valued images with vectors of any dimension. Finally we demonstrate how these algorithms generally produce better flows than the original algorithm.

Key words: Optical flow, TV, convex nonsmooth analysis, vector valued images, projections on ellipsoids, GPU implementation

1 Introduction

During the last decade estimation methods for optical flow have improved tremendously. This is in part due to ever-increasing computational power, but in particular to a wide variety of new interesting estimation methods (Xu et al. [2], Sun et al. [3], Zimmer et al. [4]) as well as novel implementation choices, that have proven to effectively increase accuracy (Sun et al. [5]). A basic framework for calculating optical flow is based on the variational TV-L^1 energy formulation. This method and and variations hereof has proved to be very effective (Bruhn et al. [6], Brox et al. [7], Zach et al. [1]). In the pure form the TV-L^1 energy consists of a term penalizing the total variation of the estimated flow, and a term encouraging data matching in term of an L^1-norm that is robust to outliers. One type of closely related energies is achieved by replacing the L^1-norms with smooth Charbonnier penalties ([6], [7]), and another variation consist in

replacing the L^1-norms by smooth Huber norms [8]. In the pure form, Zach et. al [1] were the first to solve the TV-L^1 optical flow problem using nonsmooth convex analysis.

This paper presents an algorithm for calculating the TV-L^1 optical flow between two vector valued images $I_0, I_1 : \mathbb{R}^d \to \mathbb{R}^k$, which is an extension that has not previously been done in the nonsmooth convex analysis setting. The algorithm generalizes the highly influential algorithm by Zach et al. [1], and the extension allows the use of e.g. color or gradient information when calculating the flows. This additional information improves the quality of the flow, compared to only using intensity values, but we will also show that simple and clean implementations of the presented algorithms, in a number of cases even surpass the more sophisticated TV-L^1-improved algorithm [9] on training data from the Middlebury optical flow database [10] in terms of average endpoint error. The focus of this paper is not to produce perfectly engineered algorithms to compete on the Middlebury benchmark, but to develop and explore the generalizations of an elegant optical flow algorithm. It is however the hope that the work presented here will lay the ground for competitive optical flow algorithms in the future.

The paper is organized as follows. In the next section we recall the TV-L^1 formulation for calculating optical flow. In Section 3 we introduce the tools used to solve our vectorial extension. A general algorithm and some implementation issues are discussed in Section 4, and examples are given in Section 5. We then present experimental results and comparisons on image sequences from the Middlebury database in Section 6, and finally we summarize and discuss future directions.

2 TV-L^1 optical flow of vector valued images

The recovery of motion patterns is clearly an important feature of human vision, and during the last two decades a large number of computer vision applications has become dependent on motion information. The optical flow is one way of expressing motion information, where we calculate the displacement field v between two images, I_0 and I_1. This field should minimize the difference $I_1(\boldsymbol{x} + v(\boldsymbol{x})) - I_0(\boldsymbol{x})$ while still being sufficiently regular. Here we will concentrate on the variational TV-L^1 formulation of the optical flow problem, where the optical flow is recovered as a minimizer of the energy E:

$$E(v) = \lambda \int_{\mathscr{T}} \|I_1(\boldsymbol{x} + v(\boldsymbol{x})) - I_0(\boldsymbol{x})\| \, \mathrm{d}\boldsymbol{x} + \int_{\mathscr{T}} \|\nabla v(\boldsymbol{x})\| \, \mathrm{d}\boldsymbol{x}. \tag{1}$$

The first term in this energy is the L^1-term, i.e. an integral of the Euclidian norm of the difference $I_1(\boldsymbol{x} + v(\boldsymbol{x})) - I_0(\boldsymbol{x})$, where the images are vector valued. This term builds on an assumption that image values are conserved over time, and along the motion. For grayscale images this implies that we do not have radical changes in the lighting of the scene, but for vector valued images the exact meaning will depend on the nature of the image format, as we will see in

Section 5. The second term simply penalizes the total variation of the flow, so the estimation favors smoother displacement fields.

The first step in minimizing this energy is to linearize the optical flow constraint around the point $\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}}$ for each $\boldsymbol{x}$,

$$I_1(\boldsymbol{x} + v(\boldsymbol{x})) - I_0(\boldsymbol{x}) \approx \underbrace{I_1(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}}) + J_{I_1}(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}})(v(\boldsymbol{x}) - \boldsymbol{v}_0^{\boldsymbol{x}}) - I_0(\boldsymbol{x})}_{=\rho(v)(\boldsymbol{x})}, \quad (2)$$

where J_{I_1} denotes the Jacobian of I_1. The problem is then split in two, introducing an auxiliary variable u, and the following energies are then minimized iteratively in a coarse-to-fine pyramid scheme

$$E_1(u) = \int_{\mathscr{T}} \|\nabla u(\boldsymbol{x})\| \, \mathrm{d}x + \frac{1}{2\theta} \int_{\mathscr{T}} \|v(\boldsymbol{x}) - u(\boldsymbol{x})\|^2 \, \mathrm{d}\boldsymbol{x}, \quad (3)$$

$$E_2(v) = \lambda \int_{\mathscr{T}} \|\rho(v)(\boldsymbol{x})\| \, \mathrm{d}x + \frac{1}{2\theta} \int_{\mathscr{T}} \|v(\boldsymbol{x}) - u(\boldsymbol{x})\|^2 \, \mathrm{d}\boldsymbol{x}. \quad (4)$$

Equation (3) is solved by the well known method by Chambolle (Chambolle [11], Bresson and Chan [12]) which is reproduced as Proposition 1 in [1], and this minimization will not be discussed in the present paper. Instead we will solve the minimization of (4). Since no differential of v is involved, the minimization of (4) boils down to a pointwise minimization, with $\boldsymbol{v}_0^{\boldsymbol{x}}$ and $\boldsymbol{u}(\boldsymbol{x})$ fixed, of a *strictly convex* cost function of the form

$$F(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{v} - \boldsymbol{u}_0\|^2 + \lambda\|A\boldsymbol{v} + \boldsymbol{b}\|. \quad (5)$$

When I_0 and I_1 are scalar-valued, $\boldsymbol{b}$ is real and $A : \mathbb{R}^d \to \mathbb{R}$ is the differential of I_1 computed at $\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}}$, which is a *linear form*. Because, when $A \neq 0$, $\boldsymbol{b}$ is always in the range of A, the minimization above boils down to computing the residual of a projection onto a closed and bounded line segment, this is the essence of Propositions 2 and 3 of [1] (when $A = 0$, the above minimization is of course trivial).

When, on the other hand, I_0 and I_1 take their values in $\mathbb{R}^k$, A becomes a linear map $\mathbb{R}^d \to \mathbb{R}^k$, and it may happen that $\boldsymbol{b} \in \mathbb{R}^k$ is not in the image (range) of A, even when $A \neq 0$. In this case the cost function F is smooth and can be minimized by usual variational methods. On the other hand, when $\boldsymbol{b} \in \operatorname{Im} A$, one needs to project onto an elliptic ball. This will de discussed in the next section.

3 A general minimization problem

In this section we present the tools used for solving the minimization problem (5). We recall first a few elements of convex analysis, the reader can refer to [13] for a complete introduction to convex analysis in both finite and infinite dimension. Here we will restrict ourselves to finite dimensional problems.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is one-homogeneous if $f(\lambda \boldsymbol{x}) = \lambda f(\boldsymbol{x})$, for all $\lambda > 0$. For a one-homogeneous function, it is easily shown that its Legendre-Fenchel transform

$$f^*(\boldsymbol{x}^*) = \sup_{\boldsymbol{x} \in \mathbb{R}^d} \{\langle \boldsymbol{x}, \boldsymbol{x}^* \rangle - f(\boldsymbol{x})\} \tag{6}$$

is the characteristic function of a closed convex set $\mathfrak{C}$ of $\mathbb{R}^d$,

$$d_{\mathfrak{C}}(\boldsymbol{x}^*) := f^*(\boldsymbol{x}^*) = \begin{cases} 0 & \text{if } \boldsymbol{x}^* \in \mathfrak{C}, \\ +\infty & \text{otherwise.} \end{cases} \tag{7}$$

The one-homogeneous functions that will interest us here are of the form $f(\boldsymbol{x}) = \|A\boldsymbol{x}\|$ where $A : \mathbb{R}^d \to \mathbb{R}^k$ is linear, and $\|\cdot\|$ is the usual Euclidean norm of $\mathbb{R}^k$. The computation of the associated Fenchel transform involves the *Moore-Penrose pseudoinverse* $A^\dagger$ of A. We recall its construction.

The kernel (or null-space) of A, denoted $\operatorname{Ker} A$, is the vector subspace of the $\boldsymbol{v} \in \mathbb{R}^d$ for which $A\boldsymbol{v} = 0$. The image (or range) of A, denoted $\operatorname{Im} A$, is the subspace of $\mathbb{R}^k$ reached by A. The orthogonal complement of $\operatorname{Ker} A$ is denoted $\operatorname{Ker} A^\perp$. call ι the inclusion map $\operatorname{Ker} A^\perp \to \mathbb{R}^d$ and π the *orthogonal* projection $\mathbb{R}^k \to \operatorname{Im} A$. It is well known that the composition map $B = \pi \circ A \circ \iota$

$$\operatorname{Ker} A^\perp \xrightarrow{\iota} \mathbb{R}^d \xrightarrow{A} \mathbb{R}^k \xrightarrow{\pi} \operatorname{Im} A \tag{8}$$

is a linear isomorphism between $\operatorname{Ker} A^\perp$ and $\operatorname{Im} A$. The Moore-Penrose pseudoinverse $A^\dagger$ of A is defined as

$$A^\dagger = \iota \circ B^{-1} \circ \pi. \tag{9}$$

With this, the following lemma provides the Legendre-Fenchel transform of $f(\boldsymbol{x})$:

Lemma 1. *The Legendre-Fenchel tranform of $\boldsymbol{x} \mapsto \|A\boldsymbol{x}\|$ is the characteristic function $d_{\mathfrak{C}}$ of the elliptic ball $\mathfrak{C}$ given by the set of $\boldsymbol{x}$'s in $\mathbb{R}^d$ that satisfy the following conditions*

$$A^\dagger A\boldsymbol{x} = \boldsymbol{x} \tag{10}$$

$$\boldsymbol{x}^\top A^\dagger A^{\dagger\top} \boldsymbol{x} \leq 1. \tag{11}$$

From the properties of pseudoinverses, the equality $\boldsymbol{x} = A^\dagger A\boldsymbol{x}$ means that $\boldsymbol{x}$ belongs to $\operatorname{Ker} A^\perp$. In fact, $A^\dagger A$ is the orthogonal projection on $\operatorname{Ker} A^\perp$. On this subspace, $A^\dagger A^{\dagger\top}$ is positive definite and the inequality thus defines an elliptic ball.

We will not prove the lemma, but we indicate how it can be done. In the case where A is the identity $\mathbb{I}_d$ of $\mathbb{R}^d$, it is easy to show that $\mathfrak{C}$ is the unit sphere of $\mathbb{R}^d$. The case where A is invertible follows easily, while the general case follows from the latter using the structure of pseudoinverse (see [14] for instance). □

We can now state the main result which allows to generalize the TV-L^1 algorithm from [1] to calculate the optical flow between two vector valued images.

Proposition 1. Minimization of (5).

(i) In the case $\boldsymbol{b} \notin \operatorname{Im} A$, $F(\boldsymbol{v})$ is smooth. It can be minimized by usual methods.
(ii) In the case where $\boldsymbol{b} \in \operatorname{Im} A$, $F(\boldsymbol{v})$, which fails to be smooth for $\boldsymbol{v} \in \operatorname{Ker} A + A^\dagger b$, reaches its unique minimum at

$$\boldsymbol{v} = \boldsymbol{u} - \pi_{\lambda\mathfrak{C}}\left(\boldsymbol{u} + A^\dagger \boldsymbol{b}\right) \tag{12}$$

where $\pi_{\lambda\mathfrak{C}}$ is the projection onto the convex set $\lambda\mathfrak{C} = \{\lambda \boldsymbol{x},\, \boldsymbol{x} \in \mathfrak{C}\}$, with $\mathfrak{C}$ as described in Lemma 1.

To see (i), write $\boldsymbol{b}$ as $A\boldsymbol{b}_0 + \boldsymbol{b}_1$, with $\boldsymbol{b}_0 = A^\dagger \boldsymbol{b}$, $A\boldsymbol{b}_0$ being then orthogonal projection of $\boldsymbol{b}$ onto $\operatorname{Im} A$, while $\boldsymbol{b}_1$ is the residual of the projection. The assumption of (i) implies that $\boldsymbol{b}_1 \neq 0$ is *orthogonal* to the image of A. One can then write

$$\|A\boldsymbol{v} + \boldsymbol{b}\| = \|A(\boldsymbol{v} + \boldsymbol{b}_0) + \boldsymbol{b}_1\| = \sqrt{\|A(\boldsymbol{v} + \boldsymbol{b}_0)\|^2 + \|\boldsymbol{b}_1\|^2} \tag{13}$$

which is always strictly positive as $\|\boldsymbol{b}_1\|^2 > 0$, and smoothness follows.

In the situation of (ii), since $\boldsymbol{b} \in \operatorname{Im} A$, we can do the substitution $\boldsymbol{v} \leftarrow \boldsymbol{v} + A^\dagger \boldsymbol{b}$ in function (5) and the resulting function has the same form as a number of functions found in [11] and [15]. We refer the reader to them for the computation of minimizers. □

Proposition 1 generalizes Propositions 2 and 3 from [1] since, on one-dimensional spaces, elliptic balls are simply line segments. The next example demonstrate this, and the subsequent example extends to multi-dimensional values.

Example 1. Consider the minimization problem

$$\arg\min_{\boldsymbol{v}} \left(\frac{1}{2}\|\boldsymbol{v} - \boldsymbol{u}_0\|^2 + \lambda |\boldsymbol{a}^\top \boldsymbol{v} + b| \right), \qquad \lambda > 0, \tag{14}$$

where $\boldsymbol{v}, \boldsymbol{u}_0 \in \mathbb{R}^d$, $\boldsymbol{a} \in \mathbb{R}^d \setminus \{0\}$. The pseudoinverse of $\boldsymbol{v} \mapsto \boldsymbol{a}^\top \boldsymbol{v}$ is the multiplication by $\boldsymbol{a}/\|\boldsymbol{a}\|^2$. Applying Lemma 1, the set $\mathfrak{C}$ is just the line segment $[-\boldsymbol{a}, \boldsymbol{a}]$ and proposition 1 gives the solution

$$\boldsymbol{u} = \boldsymbol{u}_0 - \pi_{\lambda[-\boldsymbol{a},\boldsymbol{a}]}\left(\boldsymbol{u}_0 + \frac{b}{\|\boldsymbol{a}\|^2}\boldsymbol{a} \right), \tag{15}$$

where the projection is given by

$$\pi_{\lambda[-\boldsymbol{a},\boldsymbol{a}]}\left(\boldsymbol{u}_0 + \frac{b}{\|\boldsymbol{a}\|^2}\boldsymbol{a} \right) = \begin{cases} \lambda \boldsymbol{a} & \text{if} \quad \boldsymbol{a}^\top \boldsymbol{u}_0 + b < -\lambda \|\boldsymbol{a}\|^2 \\ -\lambda \boldsymbol{a} & \text{if} \quad \boldsymbol{a}^\top \boldsymbol{u}_0 + b > \lambda \|\boldsymbol{a}\|^2 \\ -\frac{\boldsymbol{a}^\top \boldsymbol{u}_0 + b}{\|\boldsymbol{a}\|^2}\boldsymbol{a} & \text{if} \quad |\boldsymbol{a}^\top \boldsymbol{u}_0 + b| \leq -\lambda \|\boldsymbol{a}\|^2 \end{cases}. \tag{16}$$

This is easily seen to correspond to the projection step for the TV-L^1 algorithm in [1], namely Proposition 3 with $\boldsymbol{a} = \nabla I_1$ and $b = I_1(\cdot + \boldsymbol{v}_0) - \nabla I_1 \cdot \boldsymbol{v}_0 - I_0$. ∘

Example 2. Now consider the more general minimization problem

$$\arg\min_{\boldsymbol{v}} \left(\frac{1}{2}\|\boldsymbol{v}-\boldsymbol{u}\|^2 + \lambda\|A\boldsymbol{v}+\boldsymbol{b}\| \right), \qquad \lambda > 0. \tag{17}$$

where $A \in \mathbb{R}^{k\times 2}$. If A has maximal rank (i.e. 2), then is is well known that the 2×2 matrix $C = A^\dagger A^{\dagger\top}$ is symmetric and positive definite [14]. The set $\mathfrak{C}$ is then an elliptic disc determined by the eigenvectors and eigenvalues of C. If however the matrix has two linearly dependent columns $\boldsymbol{a} \neq \boldsymbol{0}$ and $c\boldsymbol{a}$, a series of straightforward calculations give

$$\operatorname{Ker} A = \mathbb{R}\boldsymbol{y}, \quad \operatorname{Ker} A^\perp = \mathbb{R}\boldsymbol{x}, \quad \operatorname{Im} A = \mathbb{R}\boldsymbol{a} \tag{18}$$

with $\boldsymbol{x} = \frac{1}{1+c^2}(1,c)^\top$ and $\boldsymbol{y} = \frac{1}{1+c^2}(-c,1)^\top$ an orthonormal basis of $\mathbb{R}^2$, and

$$A^\dagger A^{\dagger\top} = \frac{1}{(1+c^2)^2\|\boldsymbol{a}\|^2}\begin{pmatrix} 1 & c \\ c & c^2 \end{pmatrix}. \tag{19}$$

If $c = 0$, the inequality (11) from Lemma 1, just amounts to

$$\frac{u_1^2}{\|\boldsymbol{a}\|^2} \leq 1 \iff -\|\boldsymbol{a}\| \leq u_1 \leq \|\boldsymbol{a}\| \tag{20}$$

(with $\boldsymbol{u} = (u_1,u_2)^\top$), a vertical strip, while equality (10) in Lemma 1 simply says that $u_2 = 0$, thus, setting $\gamma = \|\boldsymbol{a}\|$, $\mathfrak{C}$ is the line segment

$$[-\gamma\boldsymbol{x}, \gamma\boldsymbol{x}] \subset \mathbb{R}^2. \tag{21}$$

The case where $c \neq 0$ is identical, and obtained for instance by rotating the natural basis of $\mathbb{R}^2$ to the basis $(\boldsymbol{x},\boldsymbol{y})$. ∘

4 Implementation

In this section we will consider how to use the tools developed in the previous section to implement an algorithm for calculating the TV-L^1 optical flow between vector valued images. One particular appealing feature of the formulation we have given, is that the algorithm is essentially dimensionless, in the sense that we can produce a single implementation that can take images $I_0, I_1 : \mathbb{R}^2 \to \mathbb{R}^k$ for all values of k. This can be done since the calculations given in Example 2 only depend on k for the calculation of norms. We recall that the linearized data fidelity term is given by

$$\rho(v) = \underbrace{J_{I_1}(\boldsymbol{x}+\boldsymbol{v}_0^{\boldsymbol{x}})}_{A} v(\boldsymbol{x}) + \underbrace{I_1(\boldsymbol{x}+\boldsymbol{v}_0^{\boldsymbol{x}}) - I_0(\boldsymbol{x}) - J_{I_1}(\boldsymbol{x}+\boldsymbol{v}_0^{\boldsymbol{x}})\boldsymbol{v}_0^{\boldsymbol{x}}}_{b}, \tag{22}$$

and according to Proposition 1 there are two main situations to consider when minimizing E_1. The first situation is when $b \notin \operatorname{Im} A$, which translates to $I_1(\boldsymbol{x}+$

$\boldsymbol{v}_0^{\boldsymbol{x}}) - I_0(\boldsymbol{x}) \notin \operatorname{Im} A$, and in this situation the energy is smooth and can be minimized by usual methods (following e.g. [7]). In the alternative situation, we can minimize the energy by the projection step described in Proposition 1 (ii). In the case of images with two spatial coordinates, the calculations necessary for the projection step are done in Example 2.

A generic algorithm for the vector TV-L^1 flow is given in Algorithm 1.

```
Data: Two vector valued images I_0 and I_1
Result: Optical flow field u from I_0 to I_1
for L = L_max to 0 do
    // Pyramid levels
    Downsample the images I_0 and I_1 to current pyramid level
    for W = 0 to W_max do
        // Warping
        if I_1(x + u(x)) − I_0(x) ∈ Im J_{I_1}(x + u(x)) then
            Compute v as the minimizer of E_1, using Proposition 1 (ii) on
            current pyramid level
        else
            Compute the minimizer v by gradient descent
        end
        for I = 0 to I_max do
            // Inner iterations
            Solve (3) for u on current pyramid level
        end
    end
    Upscale flows to next pyramid level
end
```

Algorithm 1: General TV-L^1 algorithm for vector valued images.

4.1 Projections on elliptic balls

As already mentioned, the set $\mathfrak{C}$ will be an elliptic ball, and when the Jacobian J_{I_1} has full rank the ball is proper, i.e. it is not degenerated to a line segment or a point. Projecting a point outside $\mathfrak{C}$ onto the boundary ellipsoid is somewhat more complicated than projection onto a line segment, and finding efficient and precise algorithms for this is still an active area of research, [16]. We have taken the approach of doing a gradient descent on the boundary ellipsoid. This approach is not very efficient in terms of computational effort, but on the upside the algorithm is very simple, and easily implemented. The algorithm is specified in Algorithm 2, where, in order to alleviate notation, we have denoted the matrix $(J_1^\dagger J_1^{\dagger\top})(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}})$ by C. The convergence criterion of the algorithm is based on the fact that the line between the original point $\boldsymbol{w}$ and the projection $\boldsymbol{v}^n$ should be orthogonal to the boundary at $\boldsymbol{v}^n$, and if this is not achieved in 100 iterations the last value is returned.

In the situation where the Jacobian has full rank, it holds that $J_{I_1} J_{I_1}^\dagger = \mathbb{I}_d$. This means that the condition of equation (10) disappears. In addition it simplifies the expression for the point we are projecting, since the $\boldsymbol{v}_0^{\boldsymbol{x}}$'s cancel

Data: The point $\boldsymbol{w}$ and the matrix C specifying the bounding ellipsoid.
Result: The orthogonal projection of $\boldsymbol{w}$ onto the ellipsoid.
Set $\boldsymbol{v}^0 = \dfrac{\boldsymbol{w}}{\sqrt{\langle \boldsymbol{w}, C\boldsymbol{w}\rangle}}$, and let the stepsize $\tau > 0$ be small enough.
while $\boldsymbol{v}^n$ *has not converged* **do**
  $\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \tau(\boldsymbol{w} + \langle \boldsymbol{v}^n, \boldsymbol{v}^n - \boldsymbol{w}\rangle C\boldsymbol{v}^n)$ `// Gradient step`
  $\boldsymbol{v}^{n+1} = \dfrac{\boldsymbol{v}^{n+1}}{\sqrt{\langle \boldsymbol{v}^{n+1}, C\boldsymbol{v}^{n+1}\rangle}}$ `// Reprojection`
end

Algorithm 2: Projection of a point $\boldsymbol{w}$ onto an ellipsoid.

out, so the point simply becomes

$$\boldsymbol{w} = J_{I_1}^{\dagger}(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}})(I_1(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}}) - I_0(\boldsymbol{x})). \tag{23}$$

4.2 Implementation choices

In the implementations we present here it has been assumed that we are always in the situation that $I_1(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}}) - I_0(\boldsymbol{x}) \in \mathrm{Im} J_{I_1}(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}})$, and when this is not the case, we simply project $I_1(\boldsymbol{x}+\boldsymbol{v}_0^{\boldsymbol{x}})-I_0(\boldsymbol{x})$ onto the image of the Jacobian, so we never do the gradient descent step for v. The justification for this is twofold. First, if noise is small (relative to the data fidelity term), one expects that a displacement vector $\boldsymbol{v}$ satisfies the following

$$0 \approx I_1(\boldsymbol{x} + \boldsymbol{v}_0^{\boldsymbol{x}}) - I_0(\boldsymbol{x}) \approx A\boldsymbol{v} + \boldsymbol{b} \tag{24}$$

with the notations of equation (22), which means that $\boldsymbol{b}$ is "approximately" in $\mathrm{Im}\, A$. In the case where noise is considerable, this step is followed by a regularization step, which should correct for the discrepancies when noise is modest in the neighborhood. Considering however the actual case that $\boldsymbol{b}$ is not in the image of the Jacobian may improve the precision of the computed optical flow slightly.

For the implementation we use five pyramid levels with a downsampling factor of 2, and the gradients used in the projection step are estimated by bicubic lookup. For the minimization of E_2 we use forward and backward differences as suggested in [11]. Finally it should be mentioned that the minimization procedure presented in Proposition 1 is highly parallel. We have chosen to implemented the algorithm in CUDA C, so the computations can be accelerated by the hundreds of cores on modern graphics processing units.

5 Examples

In this section we will consider a number of different constancy assumptions for vector valued images. We will start with perhaps the most simple example of

this. Consider two gray-scale images $I'_0, I'_1 : \mathbb{R}^2 \to \mathbb{R}$, and let

$$I_0 = \nabla I'_0, \qquad I_1 = \nabla I'_1. \tag{25}$$

Solving (1) then corresponds to computing the flow with the gradient constancy assumption (GCA) proposed by Brox et al. [7], which will typically be more robust to illumination changes [10]. A very simple implementation of the flow algorithm from the previous section can then be done by assuming that J_{I_1} always has full rank, and then simply do the ellipse projection in each step. When the Jacobian does not have full rank, a small amount of noise is added to the entries in the matrix until the determinant is no longer zero. This approach is justified from the observation that it is relatively rare that the Hessian of the original images is zero, when the derivatives are estimated using bicubic lookup, and it is our experience that the suggested procedure does not introduce a noticeable bias in the resulting flow. In addition it has the positive side effect of slightly faster computations.

The most obvious example of a vector valued constancy assumption is constancy of RGB colors, such that $I_0, I_1 :: \mathbb{R}^2 \to \mathbb{R}^3$. Other color spaces can also be used (e.g. HSV for a more robust representation [4]). The colors provide valuable discriminative information, that should typically increase the precision of the flow compared to only using brightness values. As opposed to gradient constancy (25), it is much more common that the three RGB-layers contain exactly the same information, so the calculations should take into account the rank of the Jacobian, and do the projection step according to Example 2.

An alternative higher order constancy assumption is based on the Laplacian of individual RGB-color channels. The Laplacian has the property that it is invariant to rotation or flipping in the pixel neighborhood, and so should be better suited for these types of motion (see e.g. [17]).

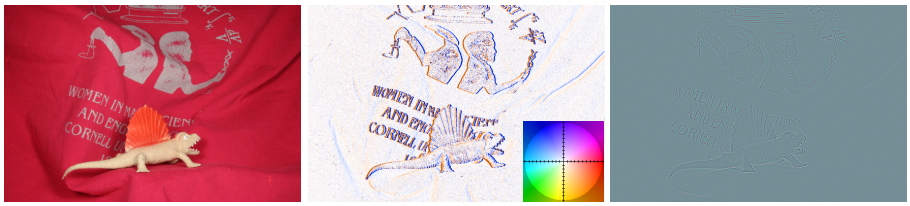


Fig. 1. Frame 10 of the *Dimetrodon* sequence represented in color, gradient (of intensities) and Laplacian of color channels. The color coding of the gradient vectors is also used for the following flow images, and the Laplacian of the color channels is represented in (rescaled) RGB.

Figure 1 contains an image from the *Dimetrodon* sequence in respectively color, gradient and Laplacian of RGB representation. The flows calculated between the RGB images and the gradient images of the Dimetrodon sequence can be seen in Figure 2. One notes that the GCA flow matches the true flow better

than the RGB version, especially along the tail of the dimetrodon. This is most likely due to the high gradients around the tip of the tail as can be seen in Figure 1. For this sequence the average endpoint errors (AEE) are 0.156 and 0.086 for the color and gradient flows respectively. The parameters used in this example can be found in Table 1.

Table 1. Parameters for the flows in Figure 2.

	warps	inner iterations	λ	θ
RGB TV-L^1	75	10	0.19	0.27
GCA TV-L^1	75	9	0.22	0.23

RGB GCA Ground truth

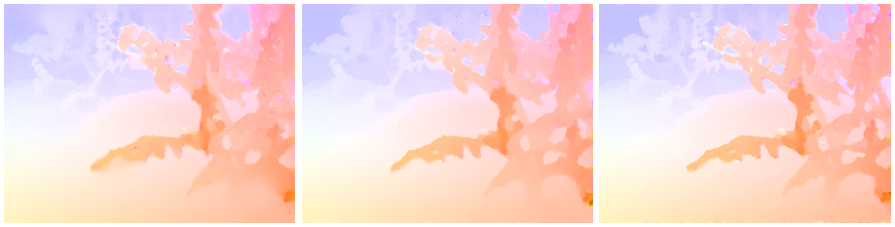
Fig. 2. Flows of the *Dimetrodon* sequence calculated using RGB TV-L^1 (AEE 0.156) and GCA TV-L^1 (AEE 0.086) respectively. Last image is ground truth from the Middlebury optical flow database.

As another example, consider the sequence *Grove3* from the Middlebury optical flow database (Figure 3). Here we are faced with a much more complicated flow pattern, and it is clear that the colors provide additional information for discriminating objects, and the flow calculated using the RGB information also results in a better flow than using just the brightness channel (Figure 4). There are however still problems with recovering the details of the small structures, which is probably due to the coarse-to-fine pyramid scheme (Xu et al. [2]).

6 Results

Results for all training sequences from the Middlebury optical flow database are available in Table 2. These results are for a fixed set of parameters for each algorithm, which can be found in Table 3. The parameters has been chosen as the ones that produce the average lowest (normalized) AAE, and have been found by an extensive grid search, with the number of warps locked at 75. The computation time for the optical flow between a pair of 640×480 RGB images is around 0.5 seconds on an NVIDIA® Tesla™ C2050 GPU for the proposed parameters, which is a factor 6 faster than the TV-L^1-improved algorithm (cf. the Middlebury database [10]). At a minor cost in accuracy (fewer warps) the RGB+MF algorithm can do realtime flow calculations for 640×480 RGB images.

Fig. 3. Frame 10 from the *Grove3* sequence and the ground truth flow.



BCA RGB RGB+MF

Fig. 4. Flows of the *Grove3* sequence calculated using brightness constancy assumption TV-L^1 (AEE 0.85), RGB TV-L^1 (AEE 0.62) and RGB TV-L^1 with a 3×3 median filtering step of the flow (AEE 0.57). The parameters are given in Table 3.

From Table 2 it can be seen that the flows calculated between gradient images will improve the results of the baseline (BCA) TV-L^1 only in a limited number of cases, however if changing lighting conditions were a bigger issue, this algorithm or the Laplacian of RGB (Δ-RGB) should be preferred.

The results for the RGB algorithm are somewhat more impressing. In six of the eight cases we see more precise flows compared to baseline, and on the two remaining sequences, the results are comparable.

Finally the results of the TV-L^1-improved algorithm from [9] and the RGB algorithm with a 3×3 median filter step are included for comparison. It should be noted that the four basic algorithms are implemented quite sparsely, i.e. without median filtering of the flow, structure–texture decomposition of the images etc., since this will correspond to minimizing an energy different from the original TV-L^1 (Sun et al. [5]). In the light of this it seems promising that the simple RGB algorithm outperforms the TV-L^1-improved on three of the eight sequences, since TV-L^1-improved uses a number of these clever tricks. We see that the addition of a small median filter improves the results of the RGB algorithm considerably, and using more of the schemes from the TV-L^1-improved algorithm will in all probability give further improvements for the algorithms presented here.

Table 2. Average endpoint error results for the Middlebury optical flow database training sequences for different constancy assumptions. Bold indicates the best result within each of the two blocks. The last two rows consist of our RGB algorithm with 3×3 median filtering and the TV-L^1-improved results from [9] for comparisson.

	BCA	GCA	RGB	Δ-RGB	RGB+MF	"improved" [9]
 Dimetrodon	0.14	**0.10**	0.16	0.22	**0.16**	0.19
 Grove2	0.18	0.23	**0.17**	0.24	**0.15**	**0.15**
 Grove3	0.85	0.76	**0.62**	0.84	**0.57**	0.67
 Hydrangea	**0.20**	0.22	0.24	0.23	0.25	**0.15**
 RubberWhale	0.20	0.20	**0.17**	0.18	0.17	**0.09**
 Urban2	0.59	0.42	**0.38**	1.52	0.36	**0.32**
Urban3	0.82	0.99	**0.62**	1.25	**0.50**	0.63
 Venus	0.54	0.58	**0.53**	0.67	0.49	**0.26**

Table 3. Global parameters for the flow results of Table 2.

	warps	inner iterations	λ	θ
BCA	75	9	0.07	0.71
GCA	75	11	0.05	0.59
RGB	75	12	0.24	0.76
Δ-RGB	75	7	0.40	0.45
RGB+MF	75	2	0.45	0.70

7 Conclusion and future research

In this paper we have proposed a generalization of the TV-L^1 optical flow algorithm by Zach et al. [1]. We have considered a number of flow algorithms based on different constancy assumptions, and it has been demonstrated that these algorithms are superior to the standard brightness constancy implementation on training data from the Middlebury optical flow database. It was even showed that some of these algorithms surpassed the more sophisticated TV-L^1-improved algorithm by Wedel et al. [9] in a number of cases. A point of interest

is to consider if further refinements from the TV-L^1-improved algorithm could also enhance the algorithms presented here. The median filter step that was included in the RGB+MF algorithm increased accuracy, as well as the robustness to wrong parameter choices, and a 5×5 median filter would probably increase accuracy even further [5]. We suspect that a structure–texture decomposition could increase the precision of the RGB TV-L^1 algorithm slightly, but the gain for the GCA TV-L^1 would probably be negligible (Sun et al. [5]). Another interesting direction would be to consider higher order data fidelity terms, e.g. GCA of RGB, GCA and RGB ($2+3$ dimensions) like in [6]. The implementation and combination of these terms is very easy in the current setup, as it can be done by simply pre-processing the input images, and inputting these new vector valued images to the same flow algorithm, similarly to what was done for the gradient constancy assumption and Laplacian of RGB. We are currently looking into these refinements, and working on implementing a competitive version of the algorithm to submit to the Middlebury optical flow database.

Another point of future research would be to automatically determine the parameters of the algorithms from the sequences. The results of Table 2 are, as already mentioned, computed from a single set of parameters, but changing the parameters can drastically improve the precision on some sequences, and degrade the quality of others (compare Figure 2 and Table 2). An excellent yet very simple approach for automatic determination of the smoothness weights is the "optimal prediction principle" proposed by Zimmer et al. [4]. An alternative approach is to define a rigorous stochastic model that allows for likelihood estimation of parameters such as the model by Markussen [18] based on stochastic partial differential equations. A step further could be to automatically determine which algorithm should be used for computing the flow, possibly only for parts of the images. One method of doing this has been successfully applied in the magnificent optical flow algorithm by Xu et al. [2], and another method was presented in [19]. In the setting of video coding, multiple motion estimates has been used in [20], where a criterion based on best interpolation quality was introduced.

Finally we are currently working on constructing specialized data fidelity terms for specific applications of optical flow, e.g. for inpainting ([21], [22]) or different schemes for variational super-resolution ([23], [24]), which should produce optical flows that are better suited for these specific tasks.

References

1. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime TV-L^1 optical flow. In: DAGM-Symposium. (2007)
2. Xu, L., Jia, J., Matsushita, Y.: A unified framework for large- and small-displacement optical flow estimation. Technical report, The Chinese University of Hong Kong (2010)
3. Sun, D., Suderth, E., Black, M.J.: Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In: NIPS. (2010)
4. Zimmer, H., Bruhn, A., Weickert, J.: Optical flow in harmony. International Journal of Computer Vision (2011)

5. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: CVPR. (2010)
6. Bruhn, A., Papenberg, N., Weickert, J.: Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In: ICCV. Volume 4. (2005) 749–755
7. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: ECCV. Volume 4. (2004) 25–36
8. Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., Bischof, H.: Anisotropic huber-l1 optical flow. In: BMVC. (2009)
9. Wedel, A., Zach, C., Pock, T., Bischof, H., Cremers, D.: An improved algorithm for TV-L^1 optical flow. In: Dagstuhl Motion Workshop. (2008)
10. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. International Journal of Computer Vision **31** (2011) 1–31
11. Chambolle, A.: An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Vision **20** (2004) 89–97
12. Bresson, X., Chan, T.: Fast dual minimization of the vectorial total variation norm and application to color image processing. Inverse Problems and Imaging **2** (2008) 455–484
13. Ekeland, I., Teman, R.: Convex Analysis and Variational Problems. SIAM (1999)
14. Golub, G., van Loan, C.: Matrix Computations. The John Hopkins University Press, Baltimore, Maryland (1989)
15. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision **40** (2011) 120–145
16. Dai, Y.H.: Fast algorithms for projection on an ellipsoid. SIAM Journal on Optimization **16** (2006) 986–1006
17. Papenberg, N., Bruhn, A., Brox, T., Didas, S., Weickert, J.: Highly accurate optical flow computations with theoretically justified warping. International Journal of Computer Vision **67** (2006) 141–158
18. Markussen, B.: Large deformation diffeomorphisms with application to optic flow. Computer Vision and Image Understanding **106** (2007) 97 – 105 Special issue on Generative Model Based Vision.
19. Mac Aodha, O., Brostow, G.J., Pollefeys, M.: Segmenting video into classes of algorithm-suitability. In: CVPR. (2010)
20. Huang, X., Rakêt, L.L., Luong, H.V., Nielsen, M., Lauze, F., Forchhammer, S.: Multi-hypothesis transform domain wyner-ziv video coding including optical flow (submitted). In: MMSP. (2011)
21. Lauze, F., Nielsen, M.: On variational methods for motion compensated inpainting. Technical report, DIKU (2009)
22. Matsushita, Y., Ofek, E., Ge, W., Tang, X., Shum, H.: Full-frame video stabilization with motion inpainting. IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006) 1150–1163
23. Unger, M., Pock, T., Werlberger, M., Bischof, H.: A convex approach for variational super-resolution. In: DAGM-symposium. (2010)
24. Keller, S., Lauze, F., Nielsen, M.: Temporal super resolution using variational methods. In: High-Quality Visual Experience: Creation, Processing and Interactivity of High-Resolution and High-Dimensional Video Signals. (2010)