Efficient Nonlocal Regularization for Optical Flow

Philipp Krähenbühl and Vladlen Koltun

Stanford University
{philkr,vladlen}@cs.stanford.edu

Abstract. Dense optical flow estimation in images is a challenging problem because the algorithm must coordinate the estimated motion across large regions in the image, while avoiding inappropriate smoothing over motion boundaries. Recent works have advocated for the use of nonlocal regularization to model long-range correlations in the flow. However, incorporating nonlocal regularization into an energy optimization framework is challenging due to the large number of pairwise penalty terms. Existing techniques either substitute intermediate filtering of the flow field for direct optimization of the nonlocal objective, or suffer substantial performance penalties when the range of the regularizer increases. In this paper, we describe an optimization algorithm that efficiently handles a general type of nonlocal regularization objectives for optical flow estimation. The computational complexity of the algorithm is independent of the range of the regularizer. We show that nonlocal regularization improves estimation accuracy at longer ranges than previously reported, and is complementary to intermediate filtering of the flow field. Our algorithm is simple and is compatible with many optical flow models.

1 Introduction

Most existing algorithms for dense motion estimation in images attempt to minimize an energy function of the form

$$E(\mathbf{u}) = E_{\text{data}}(\mathbf{u}) + \lambda E_{\text{reg}}(\mathbf{u}),$$

where $E_{\text{data}}(\mathbf{u})$ is a data term that aims to preserve image features under the motion $\mathbf{u}$, and $E_{\text{reg}}(\mathbf{u})$ is a regularization term that favors smoothness [18,4].

The data term commonly penalizes differences between image features, such as brightness [8,5], gradient or higher-order derivatives [11], or more sophisticated patch-based features [17,23]. A variety of penalty functions can be used, including the L^2 norm [8], the Lorentzian [5], the Charbonnier penalty [7], the L^1 norm [25,21], and generalized Charbonnier penalties [18]. Both the features and the penalty function can be learned from data [19].

The regularization term generally takes the form

$$E_{\text{reg}}(\mathbf{u}) = \sum_{\{i,j\} \in \mathcal{N}} \rho(u_i - u_j) w_{ij},$$

where u_i and u_j are the flow vectors corresponding to pixels i and j, ρ is a penalty function, w_{ij} is an optional data-dependent weight, and $\mathcal{N}$ is the 4-connected pixel grid. A key challenge for the regularization term is to coordinate the motion across potentially large areas in the image without inappropriately smoothing over object boundaries. To this end, a variety of robust penalty functions have been explored [5,7,11,25,18], as well as anisotropic weights that attenuate the influence of the regularizer based on the content of the image [10,22,19,24].

A complementary avenue for improving the accuracy of optical flow estimation is to employ nonlocal regularization terms that coordinate the estimated motion across larger areas in the image. This can be accomplished by establishing penalty terms on non-adjacent flow vectors.

Recently, Sun et al. [18] have proposed a nonlocal regularization objective that connects each pixel to all pixels in a small neighborhood. However, since the number of pairwise terms grows rapidly with the size of the neighborhood, the nonlocal objective is not optimized directly. Independently, Werlberger et al. [23] have advocated for the use of nonlocal regularization, and extended the total variation regularization framework to incorporate nonlocal pairwise terms. However, the optimization operates on each individual pairwise term, thus its computational complexity grows quadratically with the maximal distance of the nonlocal connections.

In this paper, we describe a new optimization algorithm that can handle nonlocal regularization objectives of any given spatial extent. The algorithm accommodates a wide range of nonlocal regularization terms, including the formulations proposed in prior work [18,23]. The nonlocal regularization objective is optimized directly, in concert with the other objectives. The computational complexity of the algorithm is linear in the size of the image and is independent of the number of nonlocal connections.

Our experiments demonstrate that nonlocal regularization provides significant benefits at longer distances than was previously reported. Furthermore, we show that intermediate filtering of the flow field is complementary to direct optimization of the nonlocal objective. Our algorithm is simple to implement and is efficient without taking advantage of parallelism or advanced hardware.

2 Nonlocal model

In this paper we augment the classical optical flow model with a nonlocal regularization term:

$$E(\mathbf{u}) = E_D(\mathbf{u}) + \lambda_l E_L(\mathbf{u}) + \lambda_n \underbrace{\sum_i \sum_{j>i} w_{ij} \rho_N(u_i - u_j)}_{E_N(\mathbf{u})} . \tag{1}$$

Here u_i is the flow vector at pixel i, E_D is the data term, and E_L is the traditional grid-connected regularization term [8,5,11,25]. E_N is the nonlocal regularization objective, defined over all pairs of pixels in the image. The penalty function used by the nonlocal term is denoted by ρ_N. Our algorithm accommodates a variety of penalty functions, including the L^2 norm $\rho_N(x) = x^2$ [8], the Lorentzian $\rho_N(x) = \log(1 + x^2/2\epsilon^2)$ [5], the Charbonnier penalty $\rho_N(x) = \sqrt{x^2 + \epsilon^2}$ [7,11], and the generalized Charbonnier penalty $\rho_N(x) = (x^2 + \epsilon^2)^\alpha$ [18].

The pairwise weights w_{ij} define the effective range of influence of E_N, which is in general anisotropic and data-dependent. In our model, the weights w_{ij} are defined by a high-dimensional Gaussian kernel in an arbitrary feature space:

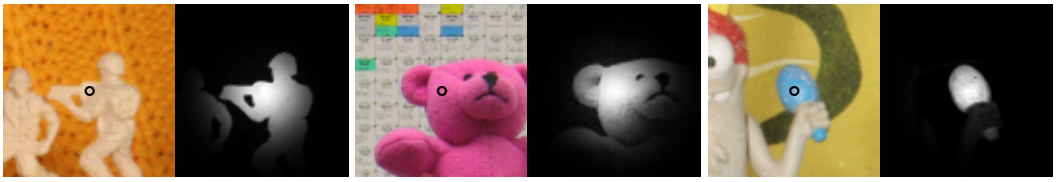
$$w_{ij} = k(\mathbf{f}_i, \mathbf{f}_j) = \exp\left(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2\sigma^2}\right),$$

where $\mathbf{f}_i$ and $\mathbf{f}_j$ are the feature vectors associated with pixels i and j. Note that this formulation allows for generalizations of the nonlocal terms used in prior work [18,23].

In our implementation, the feature vector associated with a pixel is simply a concatenation of the color and position of the pixel, and the weight is defined as

$$w_{ij} = \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\sigma_x^2} - \frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{2\sigma_c^2}\right),$$

where $\mathbf{p}_i$ is the position of pixel i and $\mathbf{c}_i$ is its color in the CIELAB color space. σ_x is the spatial standard deviation and σ_c controls color sensitivity. This definition of w_{ij} is analogous to prior



(a) Army (b) Teddy (c) Mequon

Fig. 1: Visualization of the nonlocal weights on three images from the Middlebury benchmark. In each subfigure, the greyscale image (right) visualizes the weights w_{ij} for nonlocal connections of the central pixel i with all other pixels j. Higher intensity corresponds to higher weight. The color image (left) shows the corresponding part of the original image, with the pixel i marked by a small circle.

work [18,23]. Figure 1 visualizes the nonlocal weight on three images from the Middlebury optical flow benchmark [4]. The figure shows areas of size 90×90. The spatial standard deviation σ_x was set to 15.

3 Optimization

The outer loop of our flow estimation is a standard coarse-to-fine scheme, in which the flow is estimated at increasing resolutions [5,7]. At each level, the flow is iteratively estimated by a number of warping steps. In the first iteration, the flow is estimated at the coarsest resolution directly between downsampled images. At each successive resolution, the images are warped by the previous estimate. In the remainder of this section, we focus on flow estimation during a single step of this multiresolution refinement.

At a given resolution, we estimate the flow $\mathbf{u}$ between the warped images iteratively, initializing it with the previous estimate $\mathbf{u}^0$. At step $k+1$, for $k \geq 0$, we express the flow as $\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\mathbf{u}^k$. Our goal is to find a displacement $\Delta\mathbf{u}^k$ that satisfies

$$\nabla_{\Delta\mathbf{u}^k} E(\mathbf{u}^k + \Delta\mathbf{u}^k) = 0. \tag{2}$$

Due to the complexity of the energy function E, this expression does not admit a direct solution, even if the nonlocal term E_N is not taken into account. We thus linearize the gradient around $\mathbf{u}^k$, reducing Equation 2 to a linear system [11,19]. For the data term E_D and the local regularization term E_S, the gradients are linearized as described by Papenberg et al. [11]. If the nonlocal term E_N is not present, this yields a sparse linear system that can be solved with standard techniques. We will describe how to solve (2) efficiently when the nonlocal term is present.

Our first step will be to linearize the gradient $\nabla_{\Delta\mathbf{u}^k} E_N(\mathbf{u}^k + \Delta\mathbf{u}^k)$, thus converting (2) to a linear system. Unfortunately, due to the nonlocal term, the resulting system is dense. We will use the structure of the gradient to develop a linear-time algorithm for solving the system.

To linearize the gradient ∇E_N, we begin with a simple variable substitution $\rho_N(x) = \psi_N(x^2)$. This yields the following expression for the components of the gradient:

$$\begin{aligned}\frac{\partial}{\partial \Delta u_i^k} E_N(\mathbf{u}^k + \Delta\mathbf{u}^k) = 2 \sum_{j \neq i} w_{ij} \left(u_i^k + \Delta u_i^k - u_j^k - \Delta u_j^k\right) \\ \psi_N'\left(\left(u_i^k + \Delta u_i^k - u_j^k - \Delta u_j^k\right)^2\right).\end{aligned} \tag{3}$$

Consider first the case of quadratic penalties $\rho_N(x) = x^2$. In this case, the derivative terms $\psi_N'(\cdot)$ in Equation 3 are constant ($\psi_N'(\cdot) = 1$), the gradient ∇E_N is linear in $\Delta\mathbf{u}^k$, and no further linearization is necessary. We will come back to this special case in Section 3.1.

To linearize the gradient with general penalty functions $\rho_N(x)$, we follow Papenberg et al. [11] and use a fixed point iteration to compute $\Delta\mathbf{u}^k$. We initialize this inner iteration with $\Delta\mathbf{u}^{k,0} = \mathbf{0}$. At each step $l+1$, for $l \geq 0$, we compute a displacement vector $\Delta\mathbf{u}^{k,l+1}$ that satisfies

$$\nabla_{\Delta\mathbf{u}^{k,l+1}} E(\mathbf{u}^k + \Delta\mathbf{u}^{k,l+1}) = 0, \tag{4}$$

where

$$\frac{\partial}{\partial \Delta u_i^{k,l+1}} E_N(\mathbf{u}^k + \Delta\mathbf{u}^{k,l+1}) = 2\sum_{j\neq i} w_{ij} \left(u_i^k + \Delta u_i^{k,l+1} - u_j^k - \Delta u_j^{k,l+1}\right) \psi_N'\left(\left(u_i^k + \Delta u_i^{k,l} - u_j^k - \Delta u_j^{k,l}\right)^2\right). \tag{5}$$

This assumes that the derivative terms $\psi_N'(\cdot)$ are approximately constant for small changes in the flow displacement [11]. The terms $\psi_N'(\cdot)$ in Equation 5 are constant with respect to $\Delta\mathbf{u}^{k,l+1}$, thus Equation 4 is a linear system. Specifically, we can express Equation 4 as

$$(A + B)\Delta\mathbf{u}^{k,l+1} = \mathbf{w}_A + \mathbf{w}_B, \tag{6}$$

where $B\Delta\mathbf{u}^{k,l+1} - \mathbf{w}_B$ is the sum of the linearized gradients of E_D and E_S, and A and $\mathbf{w}_A$ are defined as follows:

$$\begin{aligned} A_{ij} &= -w_{ij}\psi_N'\left(\left(u_i^k + \Delta u_i^{k,l} - u_j^k - \Delta u_j^{k,l}\right)^2\right) \qquad \text{for } i \neq j \\ A_{ii} &= \sum_{j\neq i} w_{ij}\psi_N'\left(\left(u_i^k + \Delta u_i^{k,l} - u_j^k - \Delta u_j^{k,l}\right)^2\right) \\ \mathbf{w}_A &= -A\mathbf{u}^k \end{aligned} \tag{7}$$

We solve the linear system (6) using the conjugate gradient algorithm with Jacobi preconditioning [14]. Each step of the algorithm requires multiplying a conjugate vector $\mathbf{p}$ by the matrix $A + B$. Since B is sparse, the product $B\mathbf{p}$ can be computed in time linear in the number of pixels. The matrix A, however, is dense and naive computation of the product $A\mathbf{p}$ is in general infeasible. In the remainder of this section we show that the product $A\mathbf{p}$ can be computed with a small constant number of high-dimensional Gaussian filtering operations. These operations can be performed in linear time and are extremely efficient in practice [1,9].

In Section 3.1 we introduce the algorithm in the special case of quadratic penalties. In Section 3.2 we generalize the approach to other penalty functions.

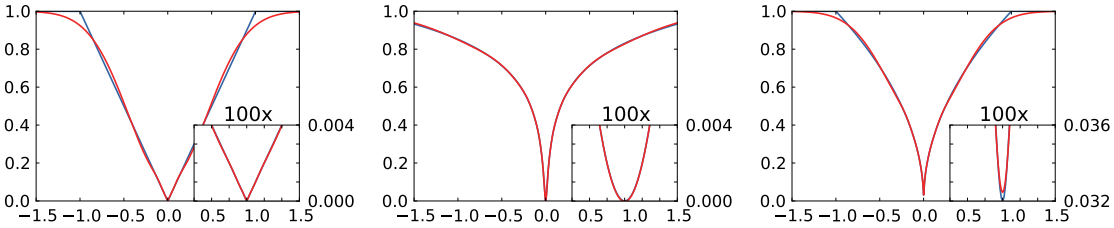
3.1 Quadratic nonlocal penalties

We first show how to efficiently compute the product $\mathbf{q} = A\mathbf{p}$, for an arbitrary vector $\mathbf{p}$, in the case of quadratic penalties $\rho_N(x) = x^2$. Each component of the product has the following form:

$$q_i = \sum_j A_{ij}p_j = p_i \sum_j k(\mathbf{f}_i, \mathbf{f}_j) - \sum_j k(\mathbf{f}_i, \mathbf{f}_j)p_j. \tag{8}$$

Here we use the definitions $w_{ij} = k(\mathbf{f}_i, \mathbf{f}_j)$ and $\psi_N'(x) = 1$. The computation of $A\mathbf{p}$ is expensive due to the summations $\sum_j k(\mathbf{f}_i, \mathbf{f}_j)$ and $\sum_j k(\mathbf{f}_i, \mathbf{f}_j)p_j$, which must be performed for each i. These summations can be written in a more general form as

$$\tilde{v}_i = \sum_j k(\mathbf{f}_i, \mathbf{f}_j)v_j, \tag{9}$$



(a) L^1 norm, $|x|$ (b) Lorentzian, $\log(1 + \frac{x^2}{2\epsilon^2})$ (c) Generalized Charbonnier, $(x^2 + \epsilon^2)^\alpha$

Fig. 2: Three penalty functions commonly used in optical flow estimation (blue) and mixtures of five exponentials fit to these functions (red). In this illustration, the functions are truncated at $T = 1$. Each inset shows a $100\times$ magnification around $x = 0$.

where $v_j = 1$ in the first summation and $v_j = p_j$ in the second.

Equation 9 describes a high-dimensional Gaussian filter, which can be evaluated for all $\tilde{v}_i$ simultaneously in linear time [12,1,9]. Our implementation uses the permutohedral lattice to perform efficient approximate high-dimensional filtering [1]. The product $A\mathbf{p}$ can thus be computed in linear time using two high-dimensional filtering operations in feature space.

3.2 General nonlocal penalties

For general nonlocal penalty functions ρ_N, the derivative ψ'_N is no longer a fixed constant. In particular, the derivative terms $\psi'_N(\cdot)$ in Equation 7 vary with the indices i, j. Thus the matrix multiplication $A\mathbf{p}$ cannot be immediately reduced to Gaussian filtering as in Section 3.1.

Approximation with exponential mixtures. To efficiently handle arbitrary penalty functions, we approximate them using mixtures of exponentials of the form $\exp(-x^2/2\sigma^2)$. Specifically, a penalty function $\rho(x)$ is approximated as

$$\rho(x) \approx \mu_{\boldsymbol{\omega},\boldsymbol{\sigma}}(x) = T - \sum_{n=1}^{K} \omega_n \exp\left(-\frac{x^2}{2\sigma_n^2}\right), \tag{10}$$

where K is the number of exponentials in the mixture, $\boldsymbol{\omega}$ is the vector of mixture weights, $\boldsymbol{\sigma}$ is the vector of exponential parameters, and T is a constant.

Exponential mixtures are closely related to Gaussian Scale Mixtures (GSM) [2,13], which have been used to approximate very general objective terms for optical flow [19]. The key difference is that we use exponential mixtures to approximate energy objectives, while GSMs have been traditionally used to model probability distributions.

Note that the range of the mixture in Equation 10 is bounded by $[T - \sum_n \omega_n, T]$. This means that we can only approximate truncated penalty functions, where T is the truncation value. In practice, this is not a limitation since we can choose a truncation value that bounds the range of our variables from above. Our implementation uses the constant $T = 50$.

We fit the parameters $\boldsymbol{\omega}, \boldsymbol{\sigma}$ of the mixture by minimizing the error

$$\int_{-\infty}^{\infty} \left(\mu_{\boldsymbol{\omega},\boldsymbol{\sigma}}(x) - \tilde{\rho}(x)\right)^2 dx, \tag{11}$$

where $\tilde{\rho}(x)$ is the truncated penalty $\tilde{\rho}(x) = \min(T, \rho(x))$. We minimize the error function using a continuous version of the Levenberg-Marquardt algorithm. This optimization procedure is

described in detail in the supplementary material. Empirically, we have found that the approximation error decreases exponentially with K. Figure 2 shows exponential mixtures fit to some of the most common penalty functions used in optical flow estimation.

Reduction to high-dimensional filtering. We now show how to perform the matrix multiplication $A\mathbf{p}$ efficiently when the penalty ρ_N is approximated by an exponential mixture $\mu_{\boldsymbol{\omega},\boldsymbol{\sigma}}(x)$. In this case, $\psi(x) = T - \sum_n \omega_n \exp(-x/2\sigma_n^2)$ and $\psi'(x) = \sum_n \frac{\omega_n}{2\sigma_n^2} \exp(-\frac{x}{2\sigma_n^2})$. Each component of the product $\mathbf{q} = A\mathbf{p}$ thus has the form

$$q_i = \sum_j A_{ij} p_j = \sum_n \frac{\omega_n}{2\sigma_n^2} \left(p_i \sum_j k(\mathbf{f}_i, \mathbf{f}_j) \exp\left(-\frac{\left(u_i^k + \Delta u_i^{k,l} - u_j^k - \Delta u_j^{k,l}\right)^2}{2\sigma_n^2} \right) - \sum_j k(\mathbf{f}_i, \mathbf{f}_j) \exp\left(-\frac{\left(u_i^k + \Delta u_i^{k,l} - u_j^k - \Delta u_j^{k,l}\right)^2}{2\sigma_n^2} \right) p_j \right). \tag{12}$$

To efficiently compute all terms q_i, consider extended feature vectors $\tilde{\mathbf{f}}_i = [\mathbf{f}_i, u_i^k + \Delta u_i^{k,l}]$ and define a new convolution kernel $\tilde{k}$ in this higher-dimensional space:

$$\tilde{k}(\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_j) = k(\mathbf{f}_i, \mathbf{f}_j) \exp\left(-\frac{\left(u_i^k + \Delta u_i^{k,l} - u_j^k - \Delta u_j^{k,l}\right)^2}{2\sigma_n^2} \right).$$

Equation 12 then takes the form

$$q_i = \sum_n \frac{\omega_n}{2\sigma_n^2} \left(p_i \sum_j \tilde{k}(\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_j) - \sum_j \tilde{k}(\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_j) p_j \right). \tag{13}$$

$\tilde{k}$ is a product of two Gaussians and is thus a Gaussian kernel in the extended feature space. Thus each of the two sums $\sum_j \tilde{k}(\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_j)$ and $\sum_j \tilde{k}(\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_j) p_j$ can be evaluated for all i in linear time, as in Section 3.1. The computation of the product $A\mathbf{p}$ reduces to $2K$ efficient high-dimensional filtering operations. For optical flow estimation, we found $K = 3$ to be sufficient.

4 Implementation

We preprocess the images by applying a structure-texture decomposition, in order to reduce the effect of illumination changes [3,21]. The structure is obtained using the Rudin-Osher-Fatemi denoising model [16]. The texture is the difference between the original image and the structure. The texture is blended with the original image (blending weight 0.05 for the original), and the blended image serves as input for optical flow estimation. This preprocessing procedure is identical to the one described by Wedel et al. [21].

We use color constancy for the data term: $E_D(\mathbf{u}) = \sum_i \rho_D(I^2(p_i + u_i) - I^1(p_i))$. Here I^1 and I^2 are the images obtained by the above preprocessing procedure, p_i is the position of pixel i, and ρ_D is the data penalty function.

We employ a standard graduated nonconvexity (GNC) scheme in the outer loop of the optimization [6,5]. We use a parameterized objective of the form $E_\alpha(\mathbf{u}) = (1-\alpha)E_Q(\mathbf{u}) + \alpha E(\mathbf{u})$, where $E_Q(\mathbf{u})$ uses quadratic penalties in all objective terms [19]. We perform three GNC steps, with $\alpha \in \{0, 0.5, 1\}$. For the first GNC step, we use the algorithm described in Section 3.1. Each subsequent GNC step is initialized with the flow computed in the previous step.

For coarse-to-fine estimation, we use a downsampling factor of 0.5 in the first GNC step and 0.8 in subsequent steps. We scale the spatial standard deviation σ_x of the nonlocal regularizer at

Method	w/o MF	MF	WMF	Runtime (sec)
Horn-Schunck	0.426	0.383	0.279	54
Horn-Schunck + NL	0.390	0.297	0.254	123
Lorentzian	0.406	0.346	0.261	173
Lorentzian + NL	0.358	0.304	0.249	678
Charbonnier	0.307	0.292	0.222	173
Charbonnier + NL	**0.270**	**0.252**	0.210	678
Generalized Charbonnier	0.307	0.287	0.222	173
Generalized Charbonnier + NL	0.272	**0.252**	**0.208**	678

Table 1: Average endpoint error on the Middelbury training set, with different penalty functions and with or without nonlocal regularization. Each model was optimized without intermediate filtering of the flow field (w/o MF), with a median filtering step after each warping iteration (MF), and with weighted median filtering (WMF). Runtime is reported for WMF and is averaged over the training set. Experiments were performed on a single CPU core.

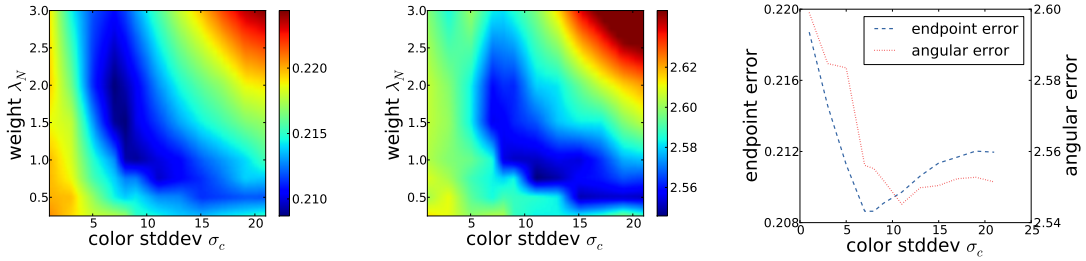
each resolution, to keep a fixed nonlocal range of influence throughout the optimization. We use three warping iterations per resolution. After each warping step, we can use a median filter to remove outliers, as proposed by Wedel et al. [21]. In Section 5, we evaluate three variants of the optimization procedure: without a median filter, with a median filter [21], and with a weighted median filter [18].

There are two common ways to formulate regularization terms for optical flow. One approach penalizes the horizontal and vertical components of the flow field separately [5,15,19,18]. The other approach penalizes differences between complete flow vectors [11,7]. We found the former approach to work better in practice and use it in all our experiments.

5 Results

Experiments were performed on the Middlebury optical flow benchmark [4], using a single core of an Intel i7-930 CPU clocked at 2.8GHz. We evaluated several versions of the model (1), using different penalty functions: the quadratic penalty $\rho(x) = x^2$ [8], the Lorentzian $\rho(x) = \log\,(1 + x^2/2\epsilon^2)$ [5], the Charbonnier $\rho(x) = \sqrt{x^2 + \epsilon^2}$ [7,11], and a generalized Charbonnier penalty $\rho(x) = (x^2 + \epsilon^2)^\alpha$ [18]. For each type of penalty, we evaluated the model with and without the nonlocal term. For each model, we evaluated three variants of the optimization procedure: without a median filtering step, with an unweighted median filter [21], and with a weighted median filter [18]. For the Charbonnier penalties we used $\epsilon = 0.001$, for the generalized Charbonnier we used $\alpha = 0.45$, and for the Lorentzian we used $\epsilon = 1.5$ for the data term and $\epsilon = 0.03$ for the regularization terms. The nonlocal penalty terms were approximated with mixtures of $K = 3$ exponentials. Table 1 shows the average endpoint error achieved by the different models and optimization procedures on the Middlebury training set.

The results indicate that nonlocal regularization substantially reduces the error for all evaluated models, even when median (or weighted median) filtering is applied to the flow field at intermediate steps. The median filter and the nonlocal regularization optimize different objectives and thus complement each other. The median filter enforces a smoothness constraint independently of any energy terms, including the data term. Due to this independence it is better able to remove strong outliers in the flow field. The nonlocal regularization on the other hand finds a smooth flow field that simultaneously minimizes the data term. If an outlier has a strong local



(a) endpoint error (b) angular error (c) error with optimal λ_N

Fig. 3: (a,b) Average endpoint error and average angular error on the Middlebury training set with varying color standard deviation σ_c and nonlocal weight λ_N. (c) Average error as a function of σ_c, with optimal λ_N for each value of σ_c. The spatial standard deviation was fixed at $\sigma_x = 9$.

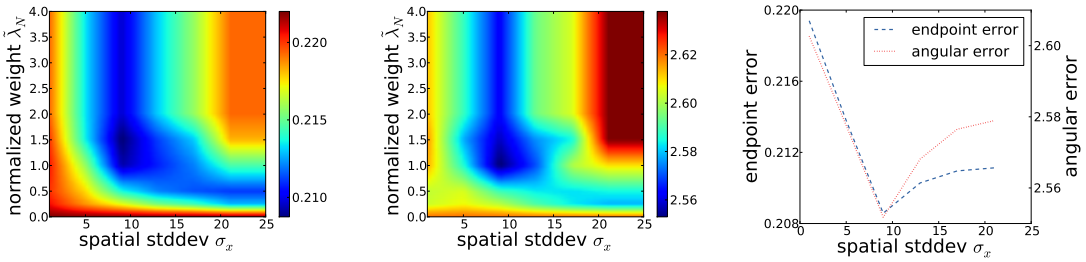
data term, neither the local nor the nonlocal regularization will be able to remove it. The combination of the two techniques is able to both remove outliers and ensure a smooth flow field that respects the data term.

The classical Horn-Schunck model with quadratic penalties performs surprisingly well when nonlocal regularization is added, even with the non-robust quadratic penalty in the nonlocal term. The Lorentzian improves on the quadratic penalty, but the improvement is much more significant for the Charbonnier and the generalized Charbonnier. This may be related to the sharp non-convexity of the Lorentzian penalty [18].

For subsequent experiments reported in this section, we use the best-performing model and optimization procedure, with generalized Charbonnier penalties, weighted median filtering, and nonlocal regularization. At the time of submission, this model is ranked 5th on both average endpoint error and average angular error on the Middlebury test set.

Nonlocal term. To evaluate the influence of the color and spatial components in the nonlocal term, we systematically varied the standard deviations σ_c and σ_x. The results are shown in Figures 3 and 4.

The color standard deviation σ_c controls the adaptivity of the nonlocal term to the image: an excessively large σ_c yields an isotropic nonlocal regularizer that does not respect object



(a) endpoint error (b) angular error (c) error with optimal λ_N

Fig. 4: (a,b) Average endpoint error and average angular error on the Middlebury training set with varying spatial standard deviation σ_x and nonlocal weight λ_N. To clearly show the behavior around the optimal parameter values, the vertical axis is scaled as a function of σ_x. Specifically, the vertical axis is parameterized by $\tilde{\lambda}_N = \lambda_N \sigma_x^2$. (c) Average error as a function of σ_x, with optimal λ_N for each value of σ_x. The color standard deviation was fixed at $\sigma_c = 8$.

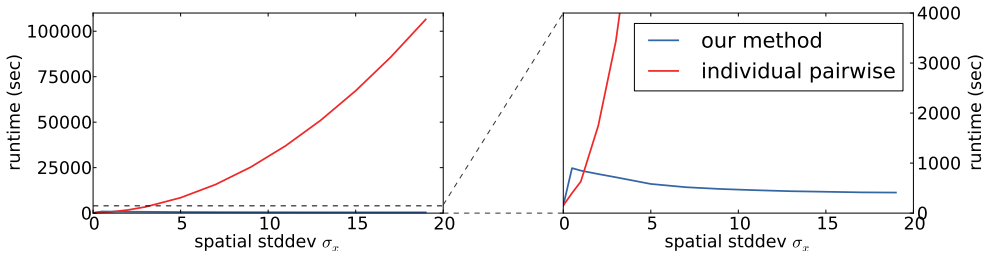


Fig. 5: Running time on the Urban dataset from the Middlebury benchmark, compared to a baseline implementation that operates on individual pairwise connections. Experiments were performed on a single CPU core.

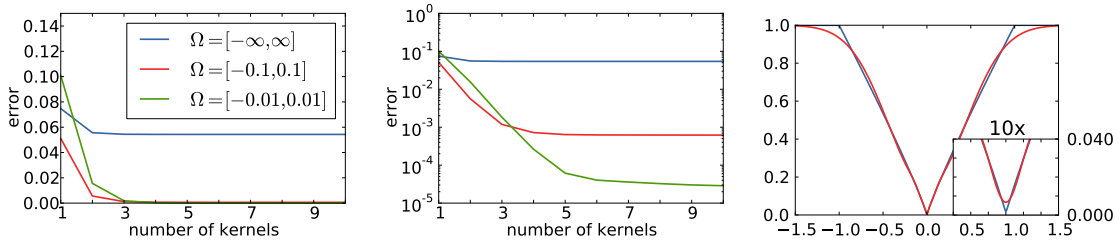
boundaries, while an excessively small σ_c yields an overly conservative regularizer that is unable to bridge small color differences. We found the optimal value to be $\sigma_c = 8$.

The spatial standard deviation σ_x controls the range of influence of the nonlocal term. We found the optimal value to be $\sigma_x = 9$. The influence of the nonlocal term is nontrivial at distances up to roughly $2\sigma_x$. (See also Figure 1.) Thus, the nonlocal regularizer connects each pixel to all pixels within a neighborhood of size roughly $4\sigma_x \times 4\sigma_x$. At $\sigma_x = 9$, the nonlocal term can establish significant connections from each pixel to roughly 1000 other pixels. Thus even if the influence of the nonlocal term is truncated at distance $2\sigma_x = 18$, the matrix A in Section 3 can have roughly $10^3 N$ nonzero entries, where N is the number of pixels in the image.

Running time. Table 1 shows the average running time of our algorithm on the Middlebury training set with and without nonlocal regularization. For the Horn-Schunck model with quadratic penalties, the running time with nonlocal regularization is roughly 2.3 times higher than the running time for the local model. This reflects the overhead of performing two high-dimensional Gaussian filtering operations per optimization step. For general penalty functions, this factor grows to 4 and each optimization step involves six filtering operations.

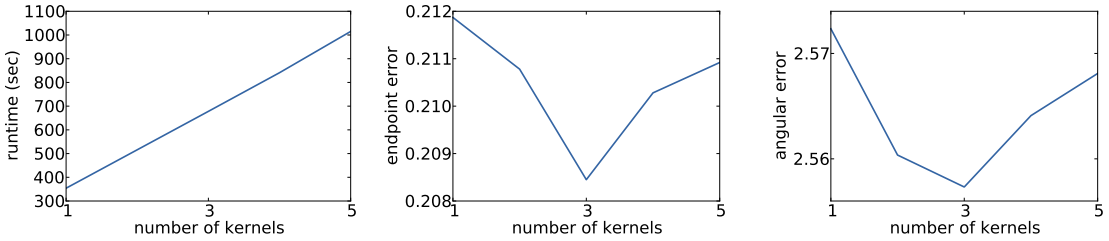
Figure 5 compares the running time of our algorithm to a straightforward implementation of the nonlocal regularizer. The baseline implementation decomposes the nonlocal term into a collection of sparse matrices. For the baseline implementation, we truncate the nonlocal term at $2\sigma_x$, limiting its influence to a patch of size $(4\sigma_x + 1) \times (4\sigma_x + 1)$ around each pixel. We also optimized the baseline implementation by disregarding pairwise connections with low weight w_{ij}. The running time of the baseline implementation grows quadratically with σ_x and is 7 hours for $\sigma_x = 9$, compared to 8 minutes for our algorithm. For $\sigma_x = 17$, the baseline implementation takes 24 hours, while our algorithm runs in 7 minutes. The gradual improvement in the running time of our algorithm with the growth of σ_x is due to the increased efficiency of high-dimensional Gaussian filtering for larger kernel sizes. Intuitively, since the maximal frequency component of the convolved function is lower, the function can be reconstructed from a sparser set of samples.

Exponential mixture approximation. We now evaluate the effect of the exponential mixture approximation described in Section 3.2. Given a penalty function $\rho(x)$ and an exponential mixture $\mu(x)$, we define the average approximation error over a domain Ω as $\frac{1}{|\Omega|} \int_\Omega |\rho(x) - \mu(x)| dx$. Figure 6 shows the average approximation error for K-kernel exponential mixtures fit to the generalized Charbonnier penalty truncated at $T = 1$. The mixtures were fit to minimize the error over the entire real line (Equation 11). The approximation error decreases exponentially with the number of kernels until it stabilizes at a certain value. Most of the error lies close to the truncation boundary. Over the real line $\Omega = \mathbb{R}$, the approximation error reaches 5.4×10^{-2} with 2 kernels



(a) approximation error (b) approximation error (log scale) (c) 3-kernel approximation

Fig. 6: (a) Average approximation error for a K-kernel exponential mixture fit to a generalized Charbonnier penalty, as a function of K. (b) Average approximation error on a log scale. (c) A 3-kernel approximation to the generalized Charbonnier penalty truncated at $T = 1$.



(a) runtime (b) endpoint error (c) angular error

Fig. 7: Effect of the number of exponential functions used to approximate the nonlocal penalty on (a) the running time of the algorithm, (b) average endpoint error on the Middlebury training set, and (c) average angular error.

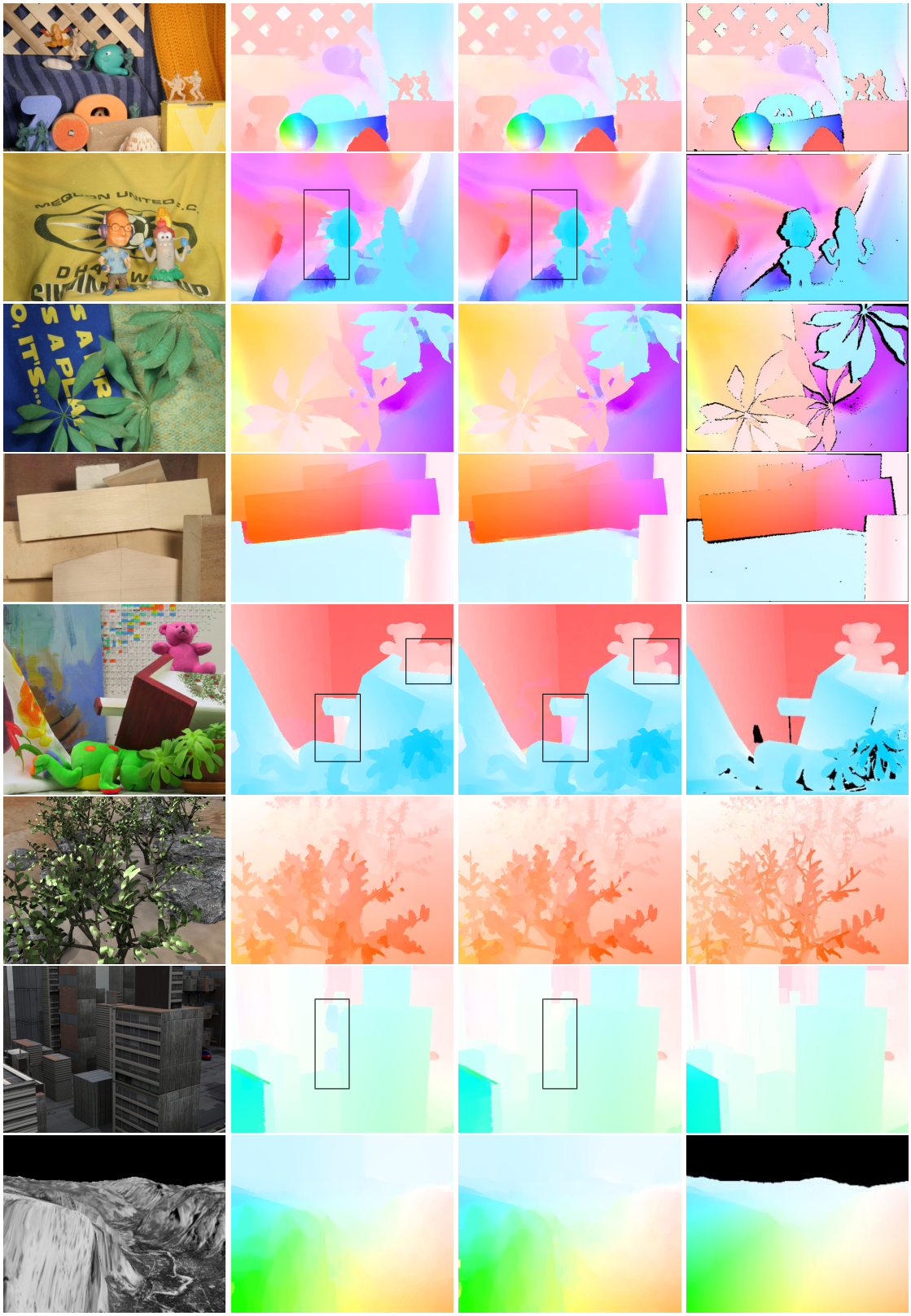
and stabilizes around that value. For $\Omega = [-0.1, 0.1]$ the approximation error stabilizes around 6×10^{-4}, and for $\Omega = [-0.01, 0.01]$ the approximation error drops to 6×10^{-5} with 5 kernels.

Figure 7 shows the effect of the number of kernels K on the performance of our algorithm. As shown in Figure 7a, the running time increases linearly with K. As shown in Figure 7b, the average endpoint error is very low even for a 1-kernel approximation to the nonlocal penalty term, suggesting that the overall v-shape of the nonlocal penalty is more important than its exact form. The error is minimized with 3 kernels and increases slightly with $K > 3$. This is likely due to the sharp minimum of the generalized Charbonnier penalty, which favors regions of constant flow. As observed by Werlberger et al. [23], rounding this peak allows for smoother flow. This is the effect achieved by the 3-kernel approximation, as shown in Figure 6c.

6 Conclusion

We presented an efficient optimization algorithm for optical flow models with nonlocal regularization. The computational complexity of the algorithm is linear in the size of the image and is independent of the number of nonlocal connections. The algorithm is simple to implement and can be parallelized for further performance gains [25,21].

We believe that the presented techniques can enhance other models for optical flow estimation. In particular, it would be interesting to integrate nonlocal regularization with high-performing layer-based approaches [20]. The integration of more robust data terms such as patch-based normalized cross correlation can lead to further improvements in accuracy [23].



(a) Image (b) Generalized Charbonnier (c) Generalized Charbonnier + NL (d) Ground truth

Fig. 8: Optical flow estimation on the Middlebury test set. Nonlocal regularization visibly improves estimation accuracy in the demarcated regions.

References

1. A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2), 2010.
2. D. F. Andrews and C. L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(1):99–102, 1974.
3. J.-F. Aujol, G. Gilboa, T. Chan, and S. Osher. Structure-texture image decomposition–modeling, algorithms, and parameter selection. *International Journal of Computer Vision*, 67:111–136, 2006.
4. S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.
5. M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.
6. A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.
7. A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.
8. B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
9. P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Proc. NIPS*, 2011.
10. 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*, 8(5):565–593, 1986.
11. N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67(2):141–158, 2006.
12. S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81(1):24–52, 2009.
13. J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003.
14. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.
15. S. Roth and M. J. Black. On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(1):33–50, 2007.
16. L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.
17. F. Steinbruecker, T. Pock, and D. Cremers. Advanced data terms for variational optic flow estimation. In *Proc. VMV*, 2009.
18. D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *Proc. CVPR*, 2010.
19. D. Sun, S. Roth, J. P. Lewis, and M. J. Black. Learning optical flow. In *Proc. ECCV*, 2008.
20. D. Sun, E. B. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *Proc. NIPS*, 2010.
21. A. Wedel, T. Pock, C. Zach, D. Cremers, and H. Bischof. An improved algorithm for TV-L^1 optical flow. In *Proc. of the Dagstuhl Motion Workshop*. Springer, 2008.
22. J. Weickert and T. Brox. Diffusion and regularization of vector- and matrix-valued images. In *Inverse Problems, Image Analysis, and Medical Imaging*, pages 251–268. AMS, 2002.
23. M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *Proc. CVPR*, 2010.
24. M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L^1 optical flow. In *Proc. BMVC*, 2009.
25. C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L^1 optical flow. In *DAGM-Symposium*, 2007.