

# Filter Flow

Steven M. Seitz\*  
University of Washington

Simon Baker  
Microsoft Research

## Abstract

The filter flow problem is to compute a space-variant linear filter that transforms one image into another. This framework encompasses a broad range of transformations including stereo, optical flow, lighting changes, blur, and combinations of these effects. Parametric models such as affine motion, vignetting, and radial distortion can also be modeled within the same framework. All such transformations are modeled by selecting a number of constraints and objectives on the filter entries from a catalog which we enumerate. Most of the constraints are linear, leading to globally optimal solutions (via linear programming) for affine transformations, depth-from-defocus, and other problems. Adding a (non-convex) compactness objective enables solutions for optical flow with illumination changes, space-variant defocus, and higher-order smoothness.

## 1. Introduction

How do two photos of the same scene differ? The camera or objects in the scene may have moved, producing optical flow. The lighting, shading, or shadows may have changed, transforming the pixel intensities. The camera or camera parameters may be different, modifying the amount of (space-varying) defocus, and altering exposure, zoom, color balance, vignetting, etc.

In this paper we introduce a framework called *filter flow* for modeling a broad class of image transformations, encompassing all of the aforementioned effects. The key idea is to model image transformations as a *space-variant* linear filter. Like a convolution, each pixel in the first image is combined with its neighborhood to produce the pixel at the same coordinates in the second image. The difference is that the filter kernel (matrix of weights) may vary from pixel to pixel. As shown in Figure 1, this model supports motion (optical flow), intensity changes, blur, and combinations of these effects.

In addition to being very general, filter flow is attractive because the transformation is *linear*. Hence, the in-

\*This work was conducted while the first author was a Consulting Researcher at Microsoft Research.

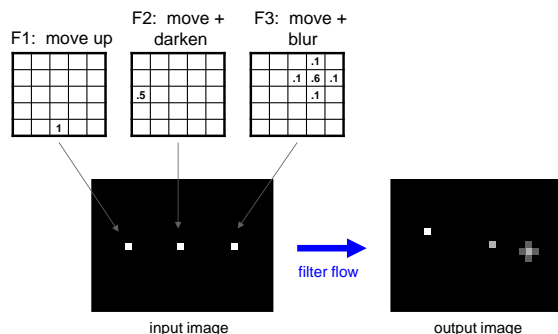


Figure 1. Filter flow generalizes optical flow by applying a *space-variant* filter at each pixel, allowing very general transformations.

verse problem of computing a transformation (i.e., per-pixel kernels) from the two images is also linear (e.g., via pseudo-inverse). The inverse problem, however, is massively underconstrained—there are many different transformations that produce a given image pair. Computing the correct transformation therefore requires identifying additional constraints on the filters that enable a unique solution without sacrificing too much expressive power.

This paper presents a catalog of constraints and objectives that can be placed on the filters to solve a wide variety of computer vision problems ranging from image alignment to shape from defocus. A particular algorithm is designed by selecting a subset of these constraints and objectives. Most of the constraints and objectives are linear. Any combination of the linear constraints and objectives can be solved to obtain the globally optimal solution using Linear Programming (LP). When non-linear constraints are added, the problem can be solved using a short sequence of LPs.

After describing the framework, we present five case studies. In the first we present a linear algorithm for computing a global affine transformation. In the second we introduce a stereo algorithm that integrates higher order smoothness priors such as 2nd order smoothness [22] and pixelwise affine smoothness. In the third we show how filter flow is used to perform stereo between a pair of grey-scale images with substantial illumination variation. By imposing MRF smoothness on both the disparities and the illumination variation, we are able to recover estimates of both.

In the fourth case study we show how to compute optical flow using the filter flow framework. We present results

on the Middlebury Flow benchmark [1] and show our algorithm to be reasonably competitive with the state of the art in terms of flow accuracy. In terms of interpolation accuracy our algorithm scores second overall, at the time of submission. Adding an illumination variation model significantly improves performance on benchmark images that contain moving shadows and other photometric effects.

In the final case study we consider the problem of estimating space-variant image defocus from a pair of images. When the images are aligned, our formulation yields a globally optimal algorithm to compute the defocus kernels (whose widths correlate to depth). We then show how to model both motion and defocus, resulting in an alignment algorithm that is robust to space-varying defocus. Our algorithm is an important step toward the practical use of depth-from-defocus in less controlled settings.

## 2. Space-variant Linear Filters

We consider the class of transformations that can be written as an  $N \times N$  matrix  $\mathbf{T}$  that operates on an image  $\mathbf{I}_1$  represented as a vector of  $N$  pixels to yield an image  $\mathbf{I}_2$ :

$$\mathbf{I}_2 = \mathbf{T}\mathbf{I}_1. \quad (1)$$

Each row of  $\mathbf{T}$  defines a linear combination of pixels in  $\mathbf{I}_1$  that produces a single pixel at the appropriate position in  $\mathbf{I}_2$ . Another way to write Eq. (1) is in terms of filtering operations rather than a single large matrix multiplication. Specifically, denote an image  $I(u, v)$  where  $\mathbf{u} = (u, v)$  denotes pixel location. At each pixel, we define a filter  $T_{ij}^{\mathbf{u}}$  that operates on a region of the image centered at pixel  $\mathbf{u}$ :

$$I_2(u, v) = \sum_{i,j} T_{ij}^{\mathbf{u}} I_1(u + i, v + j). \quad (2)$$

Note that this is a double summation, and the domains  $i \in [-k_1, k_2]$  and  $j \in [-k_3, k_4]$  are omitted to simplify notation. Eq. (2) defines a *space-variant* linear filter. Space-variant linear filters, well-known in the shape-from-defocus literature (e.g., [4, 21]), can represent a broad class of transformations including optical flow, lighting changes, blur, and their combinations (see Figure 1).

### 2.1. Factored Filters

Rather than treat the transformations atomically, it is useful to separate the geometric and photometric components. We thus factor the transformations into a composition of a space-variant motion with a space-variant filtering step: each pixel in  $I_1$  is filtered with its neighborhood and then moved to the appropriate location in  $I_2$ . Formally,  $\mathbf{T}$  is factored into a product of a motion  $\mathbf{M}$  and a kernel  $\mathbf{K}$ :

$$\mathbf{T} = \mathbf{M}\mathbf{K}. \quad (3)$$

In the case of a pure motion,  $\mathbf{K}$  is the identity matrix. Similarly, if there is no motion,  $\mathbf{M}$  is the identity.

The transformation  $\mathbf{I}_2 = \mathbf{M}\mathbf{K}\mathbf{I}_1$  can be rewritten:

$$\mathbf{M}^{-1}\mathbf{I}_2 = \mathbf{K}\mathbf{I}_1. \quad (4)$$

Note that  $\mathbf{M}^{-1}$  encodes the motion from  $I_2$  to  $I_1$  instead of from  $I_1$  to  $I_2$ . The advantage of Eq. (4) is that it yields equations that are *linear* with respect to the unknowns of  $\mathbf{M}^{-1}$  and  $\mathbf{K}$ . We therefore solve for *inverse* motion, and redefine  $\mathbf{M}$  to be motion from  $I_2$  to  $I_1$ , yielding:

$$\mathbf{M}\mathbf{I}_2 = \mathbf{K}\mathbf{I}_1. \quad (5)$$

As with  $\mathbf{T}$ , we can alternatively define  $\mathbf{K}$  and  $\mathbf{M}$  as filters  $K_{ij}^{\mathbf{u}}$  and  $M_{ij}^{\mathbf{u}}$  and rewrite Eq. (5) as:

$$\sum_{i,j} M_{ij}^{\mathbf{u}} I_2(u + i, v + j) = \sum_{i,j} K_{ij}^{\mathbf{u}} I_1(u + i, v + j). \quad (6)$$

Note that the filter kernels  $K_{ij}^{\mathbf{u}}$  and  $M_{ij}^{\mathbf{u}}$  need not be the same size; e.g., we may want to allow large motions but limit the amount of blur. Eq. (6) can then be converted into a linear (L1) **Data Objective**:

$$\left\| \sum_{i,j} M_{ij}^{\mathbf{u}} I_2(u + i, v + j) - \sum_{i,j} K_{ij}^{\mathbf{u}} I_1(u + i, v + j) \right\|_1 \quad (\text{DO})$$

This equation is hugely underconstrained. To solve for the motion and kernel filters, additional constraints are need. In the next two sections we describe a catalog of possibilities.

## 3. Catalog of Motion Models

We first consider constraints on the motion filter.  $M_{ij}^{\mathbf{u}}$ .

### 3.1. Representing Motion

We can represent (integer-precision) optical flow in a motion filter  $M_{ij}^{\mathbf{u}}$  by the constraint that, for each pixel  $\mathbf{u}$ , exactly one element of  $M_{ij}^{\mathbf{u}}$  is 1 and the rest are 0. We can state this constraint alternatively using the following three constraints on  $M_{ij}^{\mathbf{u}}$ . The first is **Non-Negativity**:

$$M_{ij}^{\mathbf{u}} \geq 0 \mid \forall i, j. \quad (\text{POS-M})$$

(We use the name **(POS-M)** to refer to this non-negativity constraint) The second is **Sum-To-One**:

$$\sum_{i,j} M_{ij}^{\mathbf{u}} = 1. \quad (\text{SUM1-M})$$

The third is **Compactness**:

$$\sum_{i,j} M_{ij}^{\mathbf{u}} \left\| [i \ j]^T - \bar{M}^{\mathbf{u}} \right\|_2^2 = 0 \quad (\text{CPSTRICT-M})$$

where  $\|\cdot\|_2$  is the L2 norm and the **centroid** is defined:

$$\bar{M}^{\mathbf{u}} = \sum_{i,j} [i \ j]^T M_{ij}^{\mathbf{u}}. \quad (11)$$

For each pixel  $\mathbf{u}$  in  $I_1$ , note that the centroid  $\bar{M}^{\mathbf{u}}$  encodes  $\mathbf{u}$ 's displacement in  $I_2$ . I.e., the **centroid represents optical flow**. Eqs. **(CPSTRICT-M)** and **(POS-M)** require  $M_{ij}^{\mathbf{u}}$  to be zero everywhere except the centroid, and Eq. **(SUM1-M)** requires the filter to be 1 at the centroid.

The compactness constraint **(CPSTRICT-M)** is too strict for our purposes in that it does not allow subpixel motions. It is also relatively difficult to enforce in an optimization framework. Hence, we relax Eq. **(CPSTRICT-M)** with a **Compactness Objective** that we seek to minimize:

$$\sum_{i,j} M_{ij}^{\mathbf{u}} \left\| [i \ j]^T - \bar{M}^{\mathbf{u}} \right\|_2^2. \quad (\text{CP-M})$$

Eq. **(CP-M)** imposes compactness in a soft manner, the weight of which can be regulated, and can be optimized iteratively as described in Section 5. We also consider an even softer, **Soft Compactness Objective**:

$$\sum_{i,j} M_{ij}^{\mathbf{u}} \left[ \max \left( 0, \left\| [i \ j]^T - \bar{M}^{\mathbf{u}} \right\|_2^2 - c^2 \right) \right] \quad (\text{SCP-M})$$

where  $c$  is a small constant ( $c = 1$  in our experiments.) Eq. **(SCP-M)** is softer in the sense that it does not affect any pixels which are less than L2 distance  $c$  from the centroid  $\bar{M}^{\mathbf{u}}$ , useful for accurately modeling sub-pixel motions.

Applying Eqs. **(POS-M)**, **(SUM1-M)** and **(CP-M)** constrain  $M^{\mathbf{u}}$  to approximate a pure flow field. Solving for a unique motion, however, requires adding constraints or objectives on the *form* of the motion, as described next.

## 3.2. Global Motion Models

### 3.2.1 Global translations

The motion must be a global translation if we constrain the centroid of every filter to be the same as the first one using the **Global Translation Constraints**:

$$\bar{M}^{\mathbf{u}} = \bar{M}^{\mathbf{0}} \mid \forall \mathbf{u}. \quad (\text{XLATE-M})$$

The combination of **(XLATE-M)**, **(POS-M)**, **(SUM1-M)** leads to a linear algorithm to estimate a global translation.

### 3.2.2 Global affine motion

To constrain the motion to be a global affine transformation, we introduce the unknown affine parameters as a single  $2 \times 3$  matrix  $\mathbf{A}$ . The **Global Affine Constraints** are then:

$$\mathbf{A}[u \ v \ 1]^T = \bar{M}^{\mathbf{u}} \mid \forall \mathbf{u}. \quad (\text{GLOBA-M})$$

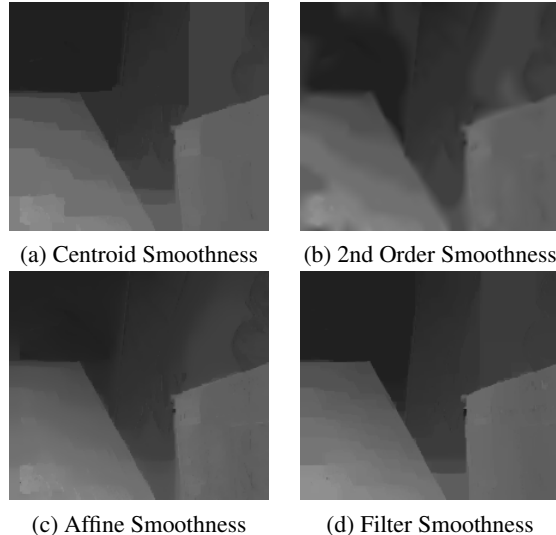


Figure 2. Comparison of MRF smoothness terms. 2nd order smoothness produces smoother disparities compared to centroid smoothness, but over-smooths discontinuities. Affine smoothness yields both smooth disparities and relatively clean discontinuities. Filter smoothness is not quite as smooth as affine, but does best at discontinuities, due to the implicit truncation. These were computed using the filter flow stereo approach presented in Section 6.2.

The affine approach is easily generalized to **polynomial** spatial transformations (e.g., radial lens distortion), where the affine matrix is replaced by a  $2 \times K$  matrix  $\mathbf{P}$  of unknown coefficients, and  $[u \ v \ 1]$  is replaced with the vector of all (a priori known)  $K$  monomial terms  $u^a v^b$ .

## 3.3. Local Motion (MRF) Models

Local motion models (i.e., spatial regularization or smoothness) are also straightforward to enforce. In particular, we support Markov Random Field (MRF) smoothness terms that are popular in optical flow and stereo. Here we consider only linear (L1) MRF objectives. Note that while this restriction eliminates useful models like error truncation [19], it also trivially enables second-order smoothness and other higher order linear relations.

### 3.3.1 Centroid smoothness

A standard smoothness condition used in optical flow and stereo is that the motion/disparity of a pixel should be similar to that of its neighbors. The L1 form of this objective in filter flow is the **Centroid Smoothness Objective**:

$$\sum_{\mathbf{u}} \sum_{\mathbf{u}' \in N(\mathbf{u})} \|\bar{M}^{\mathbf{u}} - \bar{M}^{\mathbf{u}'}\|_1 \quad (\text{MRF1-M})$$

where  $N(\mathbf{u})$  is the set of neighbors of a pixel  $\mathbf{u}$  (our experiments employ 4-connectivity). A known limitation of Eq. **(MRF1-M)**, is that it tends to produce piece-wise constant flow fields (see Figure 2(a)). In the context of stereo, it

has a *fronto-parallel* bias, which penalizes surfaces that are not parallel to the image plane.

### 3.3.2 2nd order smoothness

Penalizing the 2nd derivative of flow rather than first derivative avoids this fronto-parallel bias [22]. In filter flow, the **2nd Order Smoothness Objective** is defined as follows:

$$\sum_{\mathbf{u}', \mathbf{u}, \mathbf{u}''} \|\bar{M}^{\mathbf{u}'} - 2\bar{M}^{\mathbf{u}} + \bar{M}^{\mathbf{u}''}\|_1 \quad (\text{MRF2-M})$$

where the sum is over all sets of three consecutive horizontal or three consecutive vertical pixels. Results are shown in Figure 2(b). Eq. (MRF2-M) is much harder to optimize in the context of graph cuts because it introduces *trinary* terms (cliques of size 3), and authors have therefore proposed more complex solution methods such as QPBO and fusion moves [22]. Observe, however, that (MRF2-M) is a linear (L1) objective and is therefore straightforward to optimize within the Filter Flow framework.

### 3.3.3 Affine smoothness

While second-order smoothness produces very smooth flow fields, we have observed that it significantly over-smooths discontinuities (see Figure 2(b)). We obtain better results using *affine smoothness* [14], in which an explicit six-parameter affine transformation is estimated at each pixel, and the affine parameters are regularized within a linear (L1) MRF. I.e., the affine transformations may be different at each pixel, but encouraged to vary smoothly. A key advantage of this approach is that it leads to a pairwise (rather than trinary) objective, and is therefore better behaved around discontinuities (see Figure 2(c)).

We introduce a per-pixel  $2 \times 3$  matrix  $\mathbf{A}^{\mathbf{u}}$  and introduce **Local Affine Constraints**:

$$\mathbf{A}^{\mathbf{u}}[u \ v \ 1]^T = \bar{M}^{\mathbf{u}} \mid \forall \mathbf{u}. \quad (\text{LOCA-M})$$

We also introduce an **Affine Smoothness Objective**:

$$\sum_{\mathbf{u}} \sum_{\mathbf{u}' \in N(\mathbf{u})} \|\mathbf{A}^{\mathbf{u}} - \mathbf{A}^{\mathbf{u}'}\|_1. \quad (\text{MRFA-M})$$

Note that affine smoothness would be very difficult to solve using existing state-of-the-art MRF solvers as it depends on having hard constraints and *continuous* variables for  $\mathbf{A}$ .

### 3.3.4 Filter smoothness

Another option is to encourage neighboring pixels to have similar *filters*, rather than similar flow fields. This is accomplished via the following **Filter Smoothness Objective**:

$$\sum_{\mathbf{u}} \sum_{\mathbf{u}' \in N(\mathbf{u})} \|M^{\mathbf{u}} - M^{\mathbf{u}'}\|_1 \quad (\text{MRFF-M})$$

Eq. (MRFF-M) approximates the Potts model [3], becoming exact when  $M^{\mathbf{u}}$  is binary with a single 1. (If the 1s are in different places, MRFF-M is always 1.) Filter smoothness has a fronto-parallel bias, but the error truncation of the Potts model better preserves boundaries (see Figure 2(d)). Note, however, that Eq. (MRFF-M) also has more objective terms and is significantly (around a factor of 10 on the results in Figure 2) more expensive to optimize compared to the above alternatives.

## 3.4. Edge/Segment weighted smoothness

All of the above local smoothness objectives weight each edge in a neighborhood equally. This uniform weighting can be generalized to a gradient- or segment-based [19] weighting. We use a segment-based weighting in our optical flow results. See Section 6.4 for the details. All the other experiments use uniform weighting.

## 4. Catalog of Kernel Models

We now consider the kernel filter. This list is not comprehensive, but includes models used in our case studies, and a few others that give a broader sense of what is possible.

Since motion is already encoded in  $M^{\mathbf{u}}$ , we can enforce the constraint that the kernel filter has zero centroid using the **Zero-Centroid Constraints**:

$$\sum_{i,j} [i \ j]^T K_{ij}^{\mathbf{u}} = \mathbf{0} \mid \forall \mathbf{u}. \quad (\text{0CENT-K})$$

In some scenarios (e.g., when we know there is no “sharpening”), we can also enforce **Non-Negativity** on the kernel:

$$K_{ij}^{\mathbf{u}} \geq 0 \mid \forall i, j. \quad (\text{POS-K})$$

A **Sum-To-One** constraint may also be appropriate in cases (e.g., for a pure blur):

$$\sum_{i,j} K_{ij}^{\mathbf{u}} = 1. \quad (\text{SUM1-K})$$

Eqs. (0CENT-K), (POS-K), and/or sum-to-one (SUM1-K) are insufficient to constrain the problem sufficiently to obtain a unique solution. Solving for a unique set of filters generally requires adding constraints or objectives on the spatial *variation* of the filters, as described next.

### 4.1. Global kernel models

A space-invariant kernel, i.e., convolution, is enforced by constraining all filters to be the same. We call this constraint the **Convolution Constraint**:

$$\bar{K}^{\mathbf{u}} = \bar{K}^{\mathbf{0}} \mid \forall \mathbf{u}. \quad (\text{CONV-K})$$

More generally, we can constrain the variation of each entry of the kernel to satisfy a global parametric form. For

example, the vignetting model from [6] (assuming vignette origin at  $[0\ 0]$ ) is expressed as a **Vignette Constraint**:

$$\mathbf{V}[\|u\|^6 \|u\|^4 \|u\|^2]^T + 1 = \sum_{ij} K_{ij}^u \mid \forall \mathbf{u} \quad (\mathbf{VIG-K})$$

where  $\mathbf{V}$  is a global vector of coefficients. Eq. (**VIG-K**) is linear in all unknowns ( $\mathbf{V}$  and  $K$ ).

## 4.2. MRF kernel smoothness

For many scenes, we may assume that the kernel varies smoothly throughout most of the image, i.e., neighboring pixels have similar kernels. Accordingly, we define the L1 MRF **Kernel Smoothness Objective**:

$$\sum_{\mathbf{u}' \in \mathcal{N}(\mathbf{u})} \left\| K_{ij}^u - K_{ij}^{\mathbf{u}'} \right\|_1. \quad (\mathbf{MRF-K})$$

## 4.3. Parameterizing Symmetric Kernels

Many filters exhibit bilateral or rotational symmetry. Although it is possible to formulate constraints on a kernel filter  $K_{ij}^u$  to enforce symmetry, it is more efficient to reparameterize the filter to eliminate the redundant degrees of freedom. For rotational symmetry, we represent a 1D radial slice of the filter as a vector  $k_i^u$ . We then express each element of the 2D filter  $K_{ij}^u$  as a linear combination of the 1D filter entries  $k_i^u$ . More specifically,  $K_{ij}^u = k_d^u$ , where  $d = \sqrt{i^2 + j^2}$  (sub-pixel values of  $d$  require interpolating the vector  $k$ ). Hence, we can represent a  $K \times K$  filter with approximately  $K/2$  parameters. Note that rotationally symmetric filters automatically have zero centroid.

## 5. Optimization

Sections 3 and 4 enumerate a catalog of constraints and objectives. Almost all of these expressions are either linear, or are L1-norms of linear expressions which can be converted to a set of linear expressions through the introduction of a dummy variable [11]. The only exceptions are compactness (**CP-M**) and soft compactness (**SCP-M**). In the absence of these nonlinear objectives, any (linear) combination of the constraints and objectives is linear, convex, and can be optimized using **Linear Programming** (LP). In our experiments we used both MOSEK's interior point solver and an internal Microsoft solver also based on an interior point method. We found both algorithms to perform similarly, and both performed significantly better than the built-in Matlab LP solver, *linsolve*.

When either compactness or soft compactness is added, we solve a sequence of linear programs, each obtained by linearizing the non-linear term(s) about the previous solution. Empirically, this procedure converges after 3–4 iterations. We initialize the algorithm by omitting the nonlinear

objective(s) and solving the resulting LP. The derivative of the compactness term (**CP-M**) with respect to  $M_{ij}^u$  is:

$$D_{ij}^u = \left\| [i\ j]^T - \bar{M}^u \right\|_2^2 - 2 \sum_{a,b} M_{ab}^u \left[ i \left( a - \sum_{c,d} c M_{cd}^u \right) + j \left( b - \sum_{c,d} d M_{cd}^u \right) \right].$$

Note that if sum to one (**SUM1-M**) is enforced, the expression in the second line above reduces to 0.  $D_{ij}^u$  is computed using the estimate of  $M_{ij}^u$  in the previous iteration. The **Linearized Compactness Objective**:

$$\sum_{i,j} M_{ij}^u D_{ij}^u \quad (\mathbf{CPL-M})$$

is then added to the LP and solved. A similar derivation can be performed for soft compactness (**SCP-M**). Note that in the case of (**CP-M**), as opposed to (**SCP-M**), the procedure is a **Concave-Convex Program** which guarantees convergence to a local minimum or saddle point [23].

For moderately large motion filters the LPs can become extremely large and no longer fit in memory. For an  $N \times N$  image with an  $m \times m$  motion filter and a  $k \times k$  kernel filter, there are  $O((k^2 + m^2)N^2)$  unknowns. In addition, a reasonable initialization is required when using either of the non-convex compactness objectives (**CP-M**) or (**CP-M**). We therefore employ a coarse-to-fine strategy, via a pyramid. Each level of a traditional **Image Pyramid** is constructed by blurring and then downsampling the previous level. A second variant, which we call a **Filter Pyramid**, also blurs the image, but instead of downsampling, we double the sampling interval of a filter, e.g., replacing a filtering operation such as Eq. (2) with:

$$\sum_{i,j} T_{ij}^u I_1(u + 2^l * i, v + 2^l * j). \quad (28)$$

at level  $l$  in the pyramid. The advantage of using a **Filter Pyramid** is that we estimate the filter at each pixel in the original image. When the results are propagated down the pyramid there is therefore less blurring over edges. We use pyramids as needed based on problem complexity. No pyramids are used for affine alignment (Section 6.1) and higher order constraints on stereo (Section 6.2). For stereo with illumination change (Section 6.3) and defocus with motion (Section 6.5) we only use filter pyramids. For optical flow (Section 6.4) we use a 2-level image pyramid. For each level of the image pyramid we use a 3-level filter pyramid.

## 6. Case Studies

### 6.1. Affine Alignment: A Convex Formulation

To compute a global affine motion, we impose non-negativity (**POS-M**), sum to one (**SUM1-M**), and the global

affine constraint (**GLOBA-M**) on the motion filter. The kernel filter is set to be the identity. The only objective is the data term (**DO**). Hence, we optimize:

$$\arg \min (\mathbf{DO}) \quad \text{s.t.} \quad (\mathbf{POS-M}), (\mathbf{SUM1-M}), (\mathbf{GLOBA-M}),$$

a single LP involving only linear constraints, objectives, and L1 norms. To confirm our LP solution is correct we synthetically warped a  $150 \times 225$  pixel image with an affine warp. The motions ranged from  $-6.1$  to  $5.5$  pixels in  $x$  and  $-7.5$  to  $7.0$  pixels in  $y$ . We used a  $15 \times 17$  pixel filter. The resulting LP has 8.7M unknowns (including about 100K dummy variables) and was solved in approximately 165 minutes. All timing results are computed on a 2x Quad Core 2.5Ghz Xeon. No pyramid was used. The solution is essentially perfect, up to interpolation errors, with a maximum error across the entire image of 0.08 pixels (at one of the corners, the errors in the middle are far less.)

Our approach is unique in that it provides a *globally optimal solution* for affine motion, via linear programming. In contrast, other known techniques are nonlinear even for linear motion models. E.g., Bergen et al. [2] linearize the *image* in each iteration. Feature-based methods are linear, but assume known feature correspondence.

## 6.2. Higher Order Priors on Stereo

For dense stereo, we replace the global affine model with a local smoothness model, and restrict each motion filter to a 1D row vector whose width is the disparity search range. We found that adding the compactness objective (**CP-M**) improves results, particularly for larger images and disparity search ranges. For centroid smoothness we optimize:

$$\arg \min (\mathbf{DO}) + \alpha (\mathbf{MRF1-M}) + \beta (\mathbf{CP-M}) \quad (29)$$

$$\text{s.t.} \quad (\mathbf{POS-M}), (\mathbf{SUM1-M}) \quad (30)$$

where  $\alpha = 1.0$  and  $\beta = 4.0$  are weights that regulate the importance of the three objectives. For 2nd order smoothness we simply replace (**MRF1-M**) with (**MRF2-M**). For affine smoothness, we replace (**MRF1-M**) with (**MRFA-M**) and add the local affine constraints (**LOCA-M**). For filter smoothness we use (**MRFF-M**).

Figure 2 compares centroid smoothness, 2nd order smoothness, affine smoothness, and filter smoothness on the half size ( $217 \times 191$ ) Venus images from [19].

## 6.3. Stereo with Illumination Change

Illumination changes between images are multiplicative and can be modeled as a per-pixel *intensity scale*, represented as a  $1 \times 1$  kernel  $K^u$  in filter flow. To add the kernel filter to stereo we just add the kernel smoothness constraint  $\gamma(\mathbf{MRF-K})$  to Eq. (29) which encourages the intensity scales to vary smoothly. In our experiments we used

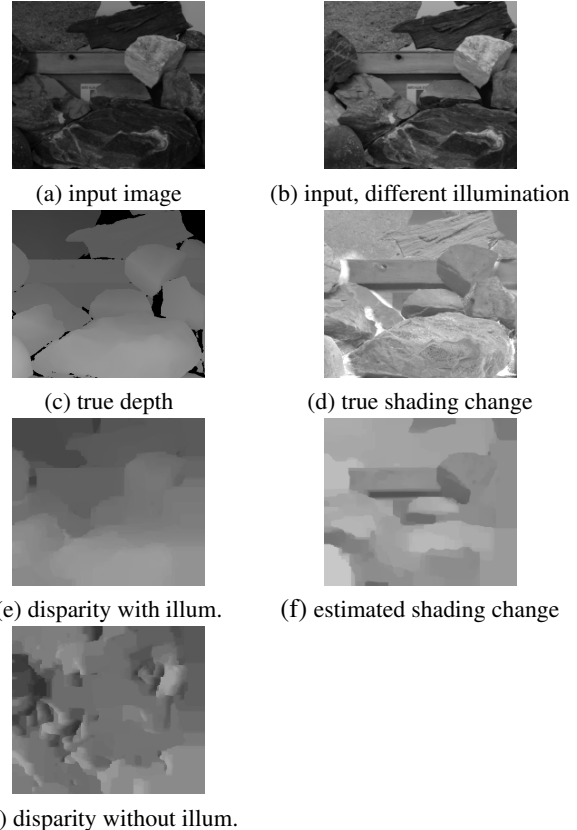


Figure 3. Stereo with illumination changes. (d) shows the shading changes (ratio of intensities) between images (a) and (b), as a result of moving the light source. Without illumination compensation, stereo performs poorly (g). Adding an MRF over shading changes greatly improves results (e), and the estimated shading (f) closely match (a regularized version of) the ground truth (d).

$\alpha = 5.0$ ,  $\beta = 1.0$ , and  $\gamma = 160$ . The results, shown in Figure 3, are computed on 1/6th size size ( $213 \times 185$ ) Rocks1 images from [9]. See Figure 3 for a description and this paper’s supplemental material for more examples.

Prior work has addressed illumination changes in stereo by use of color invariants [8] or robust matching metrics (see [9] for an excellent overview). While these techniques can be very effective, they don’t exploit the property that shading changes tend to be smooth with isolated discontinuities (object and shadow borders). Negahdaripour [13] therefore proposed to regularize intensity changes in a flow algorithm; however, his approach assumed constant parameters over a local window. Our MRF formulation globally regularizes both illumination and depth, and also produces a regularized shading change image. See Figure 3(f) for the computed shading change.

## 6.4. Optical Flow

Optical flow is formulated in the same manner as stereo, but with 2D motion filters. We found that soft compactness (**SCP-M**) produced slight more accurate subpixel flow es-

timates than compactness (**CP-M**). We also obtained better results on the Middlebury Flow benchmark [1] by adding a  $1 \times 1$  intensity-scale kernel (same scale applied to all three color channels) to model illumination changes (as in Section 6.3). In particular, intensity-scaling better handled moving shadows and other photometric effects that occur on some images in the Middlebury test set [1] (see the supplemental material for numerical results and comparison images). Note that other authors have integrated illumination modeling in optical flow (e.g., [7, 13, 20]) but did not regularize these effects in an MRF. We optimize:

$$(\mathbf{DO}) + \alpha(\mathbf{SCP-M}) + \beta(\mathbf{MRFA-M}) + \gamma(\mathbf{MRF-K})$$

s.t. (**POS-M**), (**SUM1-M**), (**LOCA-M**).

We set  $\alpha = 1.0$ ,  $\gamma = 160$ , and  $\beta = 3.5$  but also used segment based smoothness weighting where we multiply  $\beta$  by 2.5 within segments and up to 2.5 between segments depending on the average color difference; if the average colors are identical,  $\beta$  is multiplied by 2.5. We used a mean-shift-based over-segmentation algorithm, and a 2-level image pyramid, with a 3-level filter pyramid at each level of the image pyramid. For the Urban sequence (which has one of the largest motions) we used a  $8 \times 3$  pixel filter at the top level. The largest LPs at the lowest level in the image pyramid have approximately 9.2M unknowns (including around 4.6M dummy variables). The running time with the kernel filter is around 22hrs, and 9hrs without.

The results with the kernel filter are available on the Middlebury benchmark webpage [1]. In terms of flow accuracy, our algorithm is reasonably competitive with the state-of-the-art. Our results are the amongst the best to date on the difficult Urban sequence. In terms of interpolation accuracy, our algorithm is within the top two algorithms at the time of submission. Our algorithm appears to be the only one that correctly renders the orange ball in the Backyard sequence. Please see supplemental results for more details.

### 6.5. Defocus

Given a pair of images taken at different aperture or focus settings, we wish to estimate both the motion and the per-pixel defocus kernel. When the images are pre-aligned our framework provides a convex MRF formulation for the defocus problem. We set the motion filter to be the identity, use a radial parameterization of  $K^u$  (see Section 4.3), and apply MRF kernel smoothness (**MRF-K**):

$$\arg \min (\mathbf{DO}) + \alpha(\mathbf{MRF-K}) \quad \text{s.t.} \quad (\mathbf{SUM1-K}) \quad (31)$$

where  $\alpha = 7.0$ . We note that while several linear solutions to defocus have been proposed [4, 5, 15, 21], they generally assume that the scene is locally-planar so that defocus is constant over a local window. Our MRF formulation avoids this assumption, and allows global regularization.

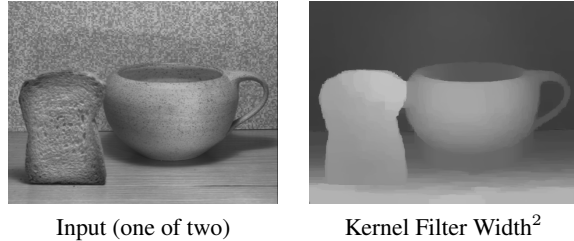


Figure 4. Defocus result on an image pair from [21] using our linear MRF-based formulation, showing estimated kernel width ( $W^2$ ) for each pixel. Light values (small  $W^2$ ) correspond to sharpening kernels, where dark values (large  $W^2$ ) are blur kernels.

We note that [12, 17] also proposed MRF frameworks for shape-from-defocus, but do not ensure a global optimum.

As our approach is non-parametric, it does **not** provide a direct measurement of *depth* (see [4] for a method to calibrate depth). For the purposes of visualization, we define the squared *width* of the kernel as:  $W^2(K_{ij}^u) = \sum_{i,j} (|i| + |j|) K_{ij}^u$  based on the fact that for a Gaussian blur  $G(\sigma)$  this quantity is proportional to  $\sigma^2$ . Because kernel width is related to depth, it provides a compelling depth visualization. One result using our linear algorithm is shown in Figure 4. This result is significantly cleaner than prior published depth from defocus results—see supplemental material for more results and comparisons.

Almost all work in depth-from-defocus assumes that the images are pre-aligned (a notable exception is [18]). Our linear formulation can be extended to include motion simply by adding terms from the motion catalog to Eq. (31). This allows us to reliably align sequences with substantial amounts of blur. In particular we conducted experiments to solve for a global affine transform to model the zoom between two images captured with different focus settings by adding the global affine constraint (**GLOBA-M**) to Eq. (31). Figure 5 shows results.

The input consists of one image focused on the front part of the carving, and the other on the back. The input images were captured with a 10MPixel digital still camera in JPG and downsampled by a factor of 10 using the Lanczos algorithm. Figure 5(c) includes the affine motion, which is roughly a zoom. Due to presence of some blur in the motion filters and subpixel interpolation in the kernel filter, we obtained better results by first using defocus-invariant affine estimation (Eq. (31) and (**GLOBA-M**)) to warp one image and then use our convex solution in Eq. (31). Our radially parameterized  $5 \times 5$  kernel filter has 3 unknowns at distance 0, 1, and 2. Pixels with distance  $> 2$  are coded to be zero. While we do not recover metric depth (although this may be possible [4]), The  $W^2$  plot (Figure 5(b)) qualitatively captures scene depth as the object recedes to the right. Note that in parts of the image, the recovered filter is a blur, and in other parts the filter is a sharpening filter with negative entries and negative  $W^2$  (Figure 5(d)).

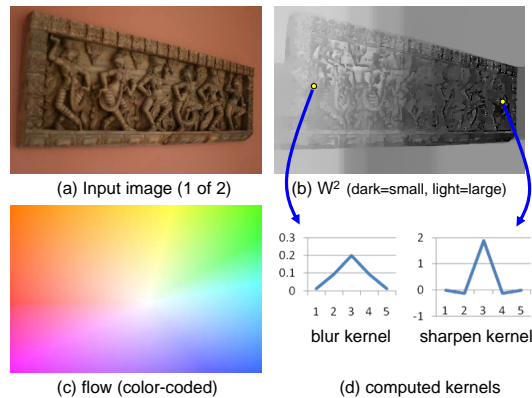


Figure 5. Two images (a) at different focus settings yield an affine flow field (c) and symmetric per-pixel kernels whose widths correlate to scene depth (b). The kernel profiles (d) correctly encode a blur to the left and a sharpen to the right.

## 7. Conclusions

This paper introduced filter flow, a framework for modeling a broad class of image transformations via space-variant filters. A major contribution of the filter flow framework, beyond its generality, is that it introduces new ways of attacking traditional motion and deconvolution problems. For example, reparameterizing optical flow from a single pair of continuous unknowns  $(f_x, f_y)$  to an  $m \times m$  motion filter leads to an optimization problem that may be solved via linear programming. While there is prior work on relaxing discrete MRFs using continuous variables [10, 16], recasting the problem into filters yields new capabilities. Filter flow has a number of advantages over prior optical flow methods, such as enabling 1) linear but non-submodular objectives (e.g., 2nd order smoothness), 2) global constraints (e.g., affine transformations), and 3) spatially-varying intensity changes and blur. Finally, we show how it is possible to derive globally optimal (convex) solutions for interesting classes of 2D motion and defocus problems.

A major limitation is speed—most results require several hours to compute. Clearly more research is needed to improve run-time. Incorporating nonlinear metrics (e.g., truncated linear error) and constraints is another important topic of future work. And finally, it would be interesting to cast several other problems within the filter flow framework, including parametric registration and fitting problems, active appearance models, and motion deconvolution.

**Acknowledgement:** we thank Sameer Agarwal for invaluable guidance on both formulation and optimization issues.

## References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *Proc. ICCV*, 2007.

[2] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. ECCV*, pages 237–252, 1992.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[4] J. Ens and P. Lawrence. An investigation of methods for determining depth from focus. *PAMI*, 15(2):97–108, 1993.

[5] P. Favaro and S. Soatto. Learning shape from defocus. In *Proc. ECCV*, pages 735–745, 2002.

[6] D. B. Goldman and J.-H. Chen. Vignette and exposure calibration and compensation. In *Proc. CVPR*, pages 899–906, 2005.

[7] H. Haussecker and D. Fleet. Computing optical flow with physical models of brightness variation. *PAMI*, 23(6):661–673, Jun 2001.

[8] Y. S. Heo, K. M. Lee, and S. U. Lee. Illumination and camera invariant stereo matching. In *Proc. CVPR*, 2008.

[9] H. Hirschmuller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *PAMI*, 31(9):1582–1599, 2009.

[10] M. Kumar, V. Kolmogorov, and P. Torr. An analysis of convex relaxations for MAP estimation. In *Proc. NIPS*, 2007.

[11] MOSEK ApS. The mosek optimization toolbox for matlab manual. [http://www.mosek.com/fileadmin/products/5\\_0/tools/doc/pdf/toolbox.pdf](http://www.mosek.com/fileadmin/products/5_0/tools/doc/pdf/toolbox.pdf), Section 6.8.3.1, Page 37.

[12] V. P. Namboodiri, S. Chaudhuri, and S. Hadap. Regularized depth from defocus. In *Proc. ICIP*, pages 1520–1523, 2008.

[13] S. Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *PAMI*, 20(9):961–979, 1998.

[14] T. Nir, A. Bruckstein, and R. Kimmel. Over-parameterized variational optical flow. *IJCV*, 76(2):205–216, 2008.

[15] A. P. Pentland. A new sense for depth of field. *PAMI*, 9(4):523–531, 1987.

[16] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers. A convex formulation of continuous multi-label problems. In *Proc. ECCV*, pages 792–805, 2008.

[17] A. N. Rajagopalan and S. Chaudhuri. Optimal recovery of depth from defocused images using an mrf model. In *Proc. ICCV*, pages 1047–1052, 1998.

[18] A. N. Rajagopalan and U. Mudenagudi. Depth estimation and image restoration using defocused stereo pairs. *PAMI*, 26(11):1521–1525, 2004.

[19] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

[20] D. Sun, S. Roth, J. P. Lewis, and M. J. Black. Learning optical flow. In *Proc. ECCV*, pages 83–97, 2008.

[21] M. Watanabe and S. Nayar. Rational Filters for Passive Depth from Defocus. *IJCV*, 27(3):203–225, 1998.

[22] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *Proc. CVPR*, 2008.

[23] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Comput.*, 15(4):915–936, 2003.