# A Coding Cost Framework for Super-resolution Motion Layer Decomposition

Thomas Schoenemann and Daniel Cremers

***Abstract*—We consider the problem of decomposing a video sequence into a superposition of (a given number of) moving layers. For this problem we propose an energy minimization approach based on coding cost. Our contributions affect both the model (*what* is minimized) and the algorithmic side (*how* it is minimized).**

**The novelty of the coding cost model is the inclusion of a refined model of the image formation process, known as super-resolution. This accounts for camera blur and area averaging arising in a physically plausible image formation process. It allows to extract sharp, high-resolution layers from the video sequence.**

**The algorithmic framework is based on an alternating minimization scheme and includes the following innovations: (1) Instead of optimizing a video labeling we optimize the layer domains. This allows to regularize the shapes of the layers and a very elegant handling of occlusions. (2) We present an efficient, parallel algorithm for extracting super-resolved layers, based on TV-filtering.**

## I. Introduction

The decomposition of videos into a superposition of moving layers is of central importance to scene interpretation, video coding and movie compression. In this paper, we present an energy minimization framework that allows to partition a given video into a set of super-resolved moving layers. Figure 1 shows an example result of this algorithm: Given an input image sequence, the algorithm reconstructs moving layers corresponding to the foreground tree and the background in a higher resolution than the individual input frames. That is, we show that by more accurately modeling the image formation process we obtain sharp, fine-detailed layer images where previous methods produced blurry ones. Moreover, we propose a graph cut based algorithm to estimate layer domains, where we show how to derive appropriate expansion moves in order to efficiently perform the layer partitioning.

### A. Related Work

Motion layer decomposition builds on a rich literature in motion analysis. We briefly sketch the main lines of research to organize the abundance of existing approaches.

*1) Motion Estimation:* Given a sequence of consecutive frames, *motion estimation* aims at computing for each frame of the sequence a velocity vector associated with each point relating it to a corresponding point in the subsequent frame. Optionally one can identify points that are occluded in the subsequent frame, yet this is rarely done.

T. Schoenemann is with the CS Department, University of Pisa, Italy. D. Cremers is with the CS Department, TU München, Germany.

input frames estimated super-resolution motion layer

Fig. 1. We propose an energy minimization approach to decompose a video into a sequence of super-resolved moving layers. In this example, a high-resolution background layer is recovered despite constant occlusion and despite a lower resolution of the individual input images.

The two major approaches to motion estimation are the local and the global one. Local approaches [24], [35], [4], [36] determine a single parameter vector to describe the motion in a fixed, usually rectangular sub-region of the image. By using overlapping regions each pixel is assigned its own velocity.

In contrast, global approaches determine a single velocity field for each frame, taking into account the entire image information at once [16]. These methods regularize the gradient of the velocity field in a variational framework. State-of-the-art methods use robust M-estimators [26], [28], [42] - a trend that arose earlier for local approaches [4]. With regularizers adapted to rigid body motion and strong contrast edges these global approaches were shown to provide some of the most accurate optic flow fields [38] on established benchmarks. The algorithms are by now rather mature, providing high-quality motion fields for $640 \times 480$ images at more than $60$ frames per second.

*2) Motion Segmentation:* By *motion segmentation* we mean the estimation of motion as well as the determination of the *boundaries* of differently moving objects. Respective algorithms have been developed in a spatially discrete MRF formulation [26] or in a spatially continuous level set formulation [9], [8]. Motion segmentation is generally considered a *chicken-and-egg-problem*: It is easy to determine an accurate segmentation for a given motion field, or vice versa to determine accurate motion models for a given segmentation. To solve for both at once has however proven to be a very difficult problem. State-of-the-art methods solve this by minimizing a single energy functional via alternating optimization schemes [3], [9], [8], [11], [32]. Precursors include methods based on pixel-flipping [27] or thresholding soft decisions [1].

Since both motion estimation and segmentation are based on intensity comparisons of consecutive frames only, they typically suffer from two limitations: Firstly, they do not exploit *long-range* temporal consistency - the fact that the

*same* intensity layer is deformed over the entire sequence is not taken into account. Secondly, they do not account for the fact that pixels may be occluded in certain frames and reappear at later stages.

*3) Layered Motion Segmentation:* Approaches for *layered motion segmentation* [40], [10] augment the framework of motion segmentation by occlusion reasoning. Dupont et al. [10] introduce a sophisticated occlusion model into traditional motion segmentation and minimize it using graph cuts and expansions moves. Xiao and Shah [40] compare each frame in the sequence to a reference frame. They introduce an occlusion order constraint which holds approximately for short sequences and – after a sophisticated initialization stage – minimize using graph cuts on three-state pixel graphs.

Although these methods improve over traditional motion segmentation, they do not fully resolve the aforementioned problems, because they are still based on comparing intensities across frames and because occlusion models are merely heuristic, not arising from a consistent image formation model.

*4) Layer Decomposition:* By *layer decomposition* we refer to the decomposition of a video sequence into a superposition of moving objects – the layers. In contrast to motion segmentation, this involves to determine the appearance of the objects in addition to their shape. Rather than an image-to-image comparison of the input video frames, we compare each video frame to a set of colored layers. Such a fully generative approach allows to model occlusions accurately.

Wang and Adelson [37] set out with the aim to decompose a sequence into a set of images. Yet, instead of treating a single model (or energy functional), they perform a multi-step optimization involving the clustering of non-parametric velocity fields. Subsequent methods managed to minimize a single energy functional [19], [39], sometimes with a sophisticated initialization [23].

Jojic and Frey [19] introduce a multi-layer decomposition method with an accurate occlusion model. They model the video as a real-valued superposition of layer images and solve this via generalized expectation maximization. Despite convincing results the method suffers from the lack of spatial smoothness and does not favor *hard* decisions to determine which video pixel belongs to which layer - two important issues to get a true decomposition. These limitations also apply to the subsequent works of Frey et al. [13] and Williams and Titsias [39] who use robust estimators.

Kumar et al. [23] propose a seven-step approach to minimize a model including motion blur and changes in lighting. This involves a combination of graph cuts and belief propagation and leads to good results for articulated motion. Yet, in this paper we show that for rigid and piecewise smooth motion one can do better. Another related approach to video decomposition for the purpose of video editing was independently and simultaneously proposed in [29].

*5) Relation to the Layer Approach of Jackson et al.:* The closest work to ours is the work of Jackson et al. [17], [18] which – regrettably – we only became aware of after publication of our CVPR paper [33]. There the authors also propose an energy minimization approach to layer decomposition. The proposed method differs from [17], [18] in several ways:

- We propose a more robust regularization of layer intensities to better preserve discontinuities in the estimation of layer intensities.
- We incorporate a super-resolution model of the layer intensity which allows for layers which are significantly sharper than both the input images and the typical layer estimates obtained by an averaging process. Solving for the intensity layers amounts to a variant of total variation deblurring. In particular, we experimentally demonstrate that sharp super-resolution layers can be estimated for numerous challenging real-world sequences.
- The proposed optimization scheme is fundamentally different. While Jackson et al. kept the layer domains fixed and thus heavily relied on the local evolution of the deformation field, we introduce graph-cut based expansion moves which allow to efficiently solve the geometric optimization problem of layer partitioning. The authors of [17], [18] admit that their approach only approximates the desired energy model (due to the regularization of the motion fields) and that they can only handle simply connected shapes. In contrast, we also estimate the layer domains and we can efficiently determine layers of arbitrary topology by means of a more global inference procedure based on expansion moves in the layer space.

### B. Contribution

We present a framework to decompose a video sequence into a given number of layers, minimizing a single energy reflecting the cost of encoding the sequence. We show how to obtain sharp, fine-detailed layer images where previous methods produced blurry ones. This is based on a physically consistent model of the image formation process – known as super-resolution – which includes camera blur and area averaging. To solve for the super-resolved layer images we apply a variant of multiple-image total variation deblurring [30], [41], [12], implemented on a GPU. To solve for the shapes of the layers we formulate a binary labeling problem on the layers.

A preliminary version of this paper appeared in [33]. The present paper contains a more in-depth discussion of both the model and the optimization algorithm. We include several novelties for the super-resolution part. In particular we show that on real-world data using absolute differences in the data term provides substantially better results than the squared differences we used in [33].

## II. From Layers to Videos and Back

We are given a video sequence consisting of $T$ frames with $X \times Y$ pixels each. This sequence is denoted

$$I : \mathbb{X} \to \mathbb{R} \quad ,$$

where $\mathbb{X} = \{1, \dots, T\} \times \{1, \dots, X\} \times \{1, \dots, Y\}$ denotes the set of spatio-temporal pixels. Layer decomposition approaches model this sequence in a generative way: as a superposition of layers moving in front of the camera. Layers are planar images of arbitrary shape and deform non-rigidly over time. The task is to infer the images together with their shapes and their motion.

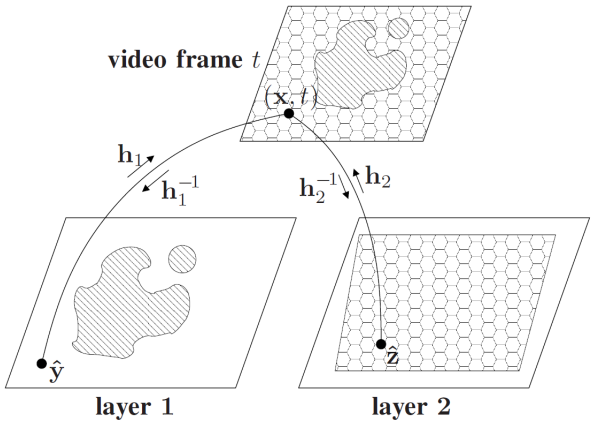



Fig. 2. Illustration of layer order (here for $N = 2$ layers), motion functions and labelings. White regions are not in the support of the layer. The following equations hold: $(\mathbf{x}, t) = \mathbf{h}_1(\hat{\mathbf{y}}, t) = \mathbf{h}_2(\hat{\mathbf{z}}, t)$ and $\hat{\mathbf{y}} = \mathbf{h}_1^{-1}(\mathbf{h}_2(\hat{\mathbf{z}}, t), t)$.

### A. Discrete vs. Continuous

An important aspect of this paper is a physically consistent model of the image formation process, which explictly states that real-world cameras produce pixel images. For this reason we choose to treat the input sequence as a discrete set of pixels. The quantities that are sought are all real-world entities and therefore modeled continuously.

### B. The Basic Setup

The aim is to represent the input sequence $I$ as a superposition of $N$ layer images

$$I_i : \Omega_i \to I\!R, \quad i = 1, \ldots, N \quad ,$$

where $N$ is given by the user. The domains $\Omega_i \subseteq I\!R^2$ are themselves unknown - they define the shapes of the layers. In the illustrative example in Figure 2 they correspond to the shaded areas.

Layers and video are from different spaces, and to improve readability we will denote points in the layer space by variables with a hat, e.g. $\hat{\mathbf{x}}$. Points in the video space are denoted without hat. Later we will represent the layer domains in terms of their characteristics functions $l_i : I\!R_2 \to \{0, 1\}$, where $l_i(\hat{\mathbf{x}}) = 1$ means that $\hat{\mathbf{x}} \in \Omega_i$. The layers are moving in front of the camera and their motion is described by functions

$$\mathbf{h}_i : \Omega_i \times [0, T) \to I\!R^2 \quad .$$

These functions describe where a layer position appears in the video at a certain time. Ideally they should be families of diffeomorphisms. That is, when fixing a time $t$ the arising function should be invertible and differentiable. We impose this condition for most of the paper, but will relax it in Section IV. Slightly abusing notation, the inverse mapping (from the video to the layer) for each time $t$ will be denoted $\mathbf{h}_i^{-1}(\cdot, t)$.

We require that $\mathbf{h}_i(\mathbf{x}, 0) = \mathbf{x}$ for all $i$, i.e. at time 0 the layer $i$ maps directly to the video, without distortion. Without this condition all entities (motion functions, layer domains and layer images) could only be determined up to a translation.

An important part of layer approaches is an occlusion model. In this work we assume that the layers are ordered and that layer $i$ occludes layer $j$ only if $i < j$. Which layer is visible at a video pixel $(\mathbf{x}, t)$ is now determined by the shapes $\Omega_i$ and the motion $\mathbf{h}_i$ of the layers. These quantities therefore induce a video labeling defining the visible layer for each image pixel:

$$l : \mathbb{X} \to \{1, \ldots, N\}.$$

$$l(\mathbf{x}, t) = \min\{i \mid \mathbf{h}_i^{-1}(\mathbf{x}, t) \in \Omega_i\} \tag{1}$$

For this expression to make sense one has to impose the constraint that the above set be non-empty:

$$\forall (\mathbf{x}, t) \in \mathbb{X} : \ \{i \mid \mathbf{h}_i^{-1}(\mathbf{x}, t) \in \Omega_i\} \neq \emptyset \quad . \tag{2}$$

In case of a violation the moving layers would not generate the video. One of the novelties of the present paper is to impose this constraint algorithmically.

## III. A Coding Cost Formulation

In this work we propose to measure the quality of a layer decomposition by the cost for encoding it. We give coding cost for two different models of the image formation process.

When imposing a layer order as in (1) a video is encoded by coding the layer domains $\Omega_i$, the intensities $I_i$ inside the layer domains and the motion functions $\mathbf{h}_i$. To get back the original input video one finally needs to code some remaining reconstruction noise, i.e. the differences between the observed and the reconstructed video. This principle is the basis of both cost functions.

### A. A Basic Coding Cost Formulation

In the first model we closely follow [17] and assume that the input images are captured by a pin-hole camera, i.e. a perfect perspective projection free of camera blur. Furthermore, the intensity of a pixel reflects a single point in the scene - in our case the respective point on the visible layer.

Under this model the intensity of a video pixel $(\mathbf{x}, t)$ is predicted by the intensity of the respective position in the visible layer $l(\mathbf{x}, t)$. To get back the original input images the remaining differences

$$I(\mathbf{x}, t) - I_{l(\mathbf{x},t)}\big(\mathbf{h}_{l(\mathbf{x},t)}^{-1}(\mathbf{x}, t)\big)$$

need to be coded. We assume that a Gaussian model – with fixed variance – gives suitable code lengths to code these differences.

To code the layer domains it suffices to encode their boundaries $\partial\Omega_i$. It is reasonable to assume that the code length increases linearly with the boundary length $|\partial\Omega_i|$. The cost of encoding the layer intensities inside the layer domains depend on the compressibility of the intensity profiles - a constant image would result in a very short code. The compressibility is well reflected by the *total variation* of the signal :

$$\int_{\Omega_i} |\nabla I_i(\hat{\mathbf{x}})| \, d\hat{\mathbf{x}} \quad . \tag{3}$$

The cost to encode the motion functions $\mathbf{h}_i$ are denoted $R(\mathbf{h}_i)$. In this paper we consider both parametric and non-parametric motion models. For most of the paper we use a

simple parametric motion model that is affine in space and quadratic in time:

$$\mathbf{h}_i(\hat{\mathbf{x}},t) = \hat{\mathbf{x}} + \mathbf{S}(\hat{\mathbf{x}},t)\,\vartheta_i, \tag{4}$$
$$\mathbf{S}(\hat{\mathbf{x}},t) = \begin{pmatrix} \hat{x}t & \hat{y}t & t & t^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \hat{x}t & \hat{y}t & t & t^2 \end{pmatrix},$$

where $\hat{\mathbf{x}} = (\hat{x}\ \hat{y})^\top$ and $\vartheta_i \in I\!R^8$ is a parameter vector. The coding cost here are negligible, so we set $R(\mathbf{h}_i) = 0$.

The parametric model satisfies the requirement of invertibility for each time $t$, but for real-world motion it is often too simple an approximation. Hence, we optionally allow to add nonparametric velocity fields to the parametric motion. As these are generally not invertible we only model the direction from video frames to layers:

$$\tilde{\mathbf{h}}_i^{-1}(\mathbf{x},t) = \mathbf{h}_i^{-1}(\mathbf{x},t) + \mathbf{v}_i^t(\mathbf{x}) \quad , \tag{5}$$

where $\mathbf{v}_i^t(\mathbf{x}) : I\!R^2 \to I\!R^2$ are nonparametric velocity fields. In analogy to the layer intensities, for this model we choose the regularity term

$$R(\tilde{\mathbf{h}}_i^{-1}) = \alpha \sum_{(\mathbf{x},t)\in\mathbb{X}} \left|\nabla \mathbf{v}_i^t(\mathbf{x})\right| \quad ,$$

where $\left|\nabla \mathbf{v}_i^t(\mathbf{x})\right|$ is a common but somewhat loose notation: to be precise one takes the gradient of each of the two components of $\mathbf{v}_i^t$ separately, then sums the absolutes of both gradients.

The arising coding cost, with weighting factors $\nu, \lambda > 0$, is

**Motion Layer Decomposition**

$$\begin{aligned} E(\{\Omega_i, I_i, \mathbf{h}_i\}) \;=\; & \sum_{(\mathbf{x},t)} \Big( I(\mathbf{x},t) - I_{l(\mathbf{x},t)}\big(\mathbf{h}_{l(\mathbf{x},t)}^{-1}(\mathbf{x},t)\big)\Big)^2 \\ & + \sum_i R(\mathbf{h}_i) \;+\; \nu \sum_i \left|\partial\Omega_i\right| \\ & + \lambda \sum_i \int_{\Omega_i} |\nabla I_i(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}} \end{aligned} \tag{6}$$

**subject to** (2)

### B. A Refined Coding Cost Formulation

In the previous section we presented a coding cost functional based on the assumption of a pin-hole camera. Minimizing the cost usually results in blurry layer images as demonstrated in Figure 3. To a large extend this is due to the inaccurate camera model: real-world cameras induce lens-blur and pixels collect the intensity inside a certain area on the sensor chip. Models which account for these effects are common in the areas of *image deblurring* and *super-resolution*. In the latter field a video sequence is reduced to a single image. In this paper we extend this idea to a superposition of layer images.

Mathematically the lens-blur is expressed as a convolution with a Gaussian kernel $b$. Its variance is set by the user. If the scene is generated by a single layer, the recorded intensities can be predicted as

$$I_{\mathrm{syn}}(\mathbf{x},t) = \int_{A(\mathbf{x})} b(\mathbf{x}') * I_1(\mathbf{h}_1^{-1}(\mathbf{x}',t))\; d\mathbf{x}' \quad , \tag{7}$$

where $A(\mathbf{x})$ is the pixel area on the sensor element. To model the input sequence as a superposition of $N$ layers we introduce a function expressing whether a layer is visible at a given video pixel or not:

$$\chi_i\big(\mathbf{x},t \,\big|\, \{\Omega_j\},\{\mathbf{h}_j\}\big) = \begin{cases} 1 \text{ if } i = \min\,\{j \,|\, \mathbf{h}_j^{-1}(\mathbf{x},t) \in \Omega_j\} \\ 0 \text{ else.} \end{cases} \tag{8}$$

The image formation process for the case of $N$ layers is now given as

$$\begin{aligned} & I_{\mathrm{syn}}(\mathbf{x},t) \\ & = \int_{A(\mathbf{x})} b(\mathbf{x}') * \left[ \sum_i \chi_i\big(\mathbf{x}',t \,\big|\, \{\Omega_j\},\{\mathbf{h}_j\}\big) I_i(\mathbf{h}_i^{-1}(\mathbf{x}',t)) \right] d\mathbf{x}' \\ & \approx \int_{A(\mathbf{x})} \sum_i \chi_i\big(\mathbf{x}',t \,\big|\, \{\Omega_j\},\{\mathbf{h}_j\}\big) \left[ b(\mathbf{x}') * I_i(\mathbf{h}_i^{-1}(\mathbf{x}',t)) \right] d\mathbf{x}' \end{aligned} \tag{9}$$

where for computational simplicity we have neglected camera blur across motion boundaries in the camera image. For the coding cost this simply implies that different difference images have to be coded:

**Super-resolution Motion Layer Decomposition**

$$\begin{aligned} E(\{\Omega_i, I_i, \mathbf{h}_i\}) = & \sum_{(\mathbf{x},t)} \left| I(\mathbf{x},t) - I_{\mathrm{syn}}(\mathbf{x},t) \right| \\ & + \sum_i R(\mathbf{h}_i) \;+\; \nu \sum_i \left|\partial\Omega_i\right| \\ & + \lambda \sum_i \int_{\Omega_i} |\nabla I_i(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}} \end{aligned} \tag{10}$$

**subject to** (2)

This time we take the Laplacian distribution to code the difference images. The reason is discussed in the following section. Minimizing the cost functional (10) results in the desired sharp, fine-detailed images, as shown in Figure 3.

### C. Discussion of the Cost Functions

We have motivated the layer decomposition functionals (6) and (10) in terms of coding cost. From a computer vision perspective there are two important points to note here.

The first point concerns the data term: for the basic functional we choose squared differences, but for the super-resolved version we use absolute differences. The reasons for this are twofold: to get a good (but slightly imperfect) notion of the layers from a poor initialization, the squared differences are better suited as the optimization algorithm is less likely to end in a poor local minimum.

However, to get fine-detailed, super-resolved layer images from a good initialization, the absolute differences are much better suited. In the experimental section we show on real-world data that absolute differences handle difficulties such as specular reflections and imprecise motion models much better than squared differences. The robustness of absolute differences to outliers is often discussed in the literature. Yet, its usefulness is usually demonstrated on synthetic data.

3/30 images | close-up of input frame | layer estimated with pin-hole camera model | layer estimated with super-resolution model

Fig. 3. Under the assumption of the pin-hole camera the extracted layers are more blurry than the input frames. In contrast, with super-resolution one recovers fine details that are not visible in any of the input frames.

The second point concerns the regularity terms in the layer space, as introduced by Jackson et al. [17]. As shown in the first author's thesis such terms are crucial to remove the otherwise occurring[1] trivial minima: often, regularity terms are skipped altogether [19], [39] or affect the length of the segmentation boundary in the video [23]. In both cases one can show that the global optimum has cost zero and is given by a single layer, obtained by unrolling the sequence - see Figure 4. Note that the associated motion function is

$$\mathbf{h}_1(\hat{\mathbf{x}}, t) = \begin{pmatrix} t\,X \\ 0 \end{pmatrix} + \hat{\mathbf{x}} \ ,$$

where $X$ is the width of the rectangular domain $\Omega$. The layer is given by

$$I_1(\begin{pmatrix} t\,X \\ 0 \end{pmatrix} + \hat{\mathbf{x}}) = I(\hat{\mathbf{x}}, t) \ .$$

## IV. Optimizing the Coding Cost

To optimize either of the coding cost (6) or (10), the algorithm must solve for the shapes (or domains) $\Omega_i$ of the layers, their appearance $I_i$ and their motion $\mathbf{h}_i$. In some cases one also wants to optimize the occlusion order. The number of layers is assumed to be given.

### A. Initialization

In the following we present an alternating optimization scheme for minimizing first the coding cost (6) and later (10). The algorithm finds a local minimum of each cost functional and is dependent on initialization. Yet, finding a good initialization is outside the scope of this paper, and it is one of our aims to show that the proposed method works even with standard initializations.

Our simple initialization scheme first chops the input video into $N$ horizontal stripes. We then initialize the motion models using the method of Lucas and Kanade [24] in each segment. This is carried out for the frames 1 and 2, assuming a translatory motion. We then recompute the visibility such that it respects the layer order, and solve for the arising layer domains and intensity profiles.

[1]This affects layer decomposition, but not layered motion segmentation.

### B. Outline of the Algorithm

To optimize the functionals (6) and (10) we use an alternating minimization framework: iteratively the motion models, the layer domains and the layer appearances are updated. The process is continued until no further energy decrease is possible, i.e. a local minimum is found.

For the layer domains and intensities globally optimal solutions are given when fixing the other quantities, the motion models are locally refined using Taylor expansions of the video intensities. To robustify the optimization we use a multi-scale scheme, starting from a coarse scale (we take 92 pixels in x-direction), which is successively enlarged (by a factor of 1.025). When reaching the full scale the alternation process is continued until a local minimum is found.

To optimize the refined cost (10) we start from the solution of the basic cost. Here we fix the motion parameters $\vartheta_i$ and only estimate the nonparametric velocity components.

## V. Alternating Minimization for the Coding Cost

We now give an in-depth description of how each quantity is updated in the alternating minimization scheme.

### A. Update of the Motion Models

For the update of motion models we distinguish two phases: for the basic cost, only the parametric model is estimated. For the refined cost this model is held fixed and optionally the nonparametric component is estimated.

*1) Updating the Parametric Models:* To update the parameters of the motion parameters, we use the relation

$$\sum_t \int_{I\!R^2} \Big( I(\mathbf{x}, t) - I_{l(\mathbf{x},t)}\big(\mathbf{h}^{-1}_{l(\mathbf{x},t)}(\mathbf{x}, t)\big)\Big)^2 \, d\mathbf{x} \tag{11}$$

$$= \sum_{i,t} \int_{I\!R^2} \chi_i\big(\mathbf{h}_i(\hat{\mathbf{x}}, t), t\big) \cdot \big(I(\mathbf{h}_i(\hat{\mathbf{x}}, t), t) - I_i\big(\hat{\mathbf{x}}\big)\big)^2 \left| \frac{d\mathbf{h}_i}{d\hat{\mathbf{x}}} \right| d\hat{\mathbf{x}} \ ,$$

where we switch from the video space to the layer space. For our discrete sensor model this holds only approximately but gives good enough results in practice. We can now apply the

input sequence

globally optimal layer when choosing inappropriate regularity terms

Fig. 4. When choosing inadequate regularity terms, layer decomposition provides meaningless solutions: the global optimum is then a single layer obtained by unrolling the sequence.

Gauss-Newton method where for a given parameter vector $\vartheta_i^0$ one performs a first-order Taylor expansion on the image:

$$\begin{aligned} &I(\hat{\mathbf{x}} + \mathbf{S}(\hat{\mathbf{x}}, t)\,(\vartheta_i^0 + \boldsymbol{\Delta}\vartheta_i), t) \\ &\quad \approx I(\hat{\mathbf{x}} + \mathbf{S}(\hat{\mathbf{x}}, t)\,\vartheta_i^0, t) + \nabla I(\hat{\mathbf{x}} + \mathbf{S}(\hat{\mathbf{x}}, t)\,\vartheta_i^0, t)^\top \boldsymbol{\Delta}\vartheta_i \quad . \end{aligned}$$

Inserting this into (11) yields a functional that is quadratic in $\boldsymbol{\Delta}\vartheta_i$, but generally not convex. To ensure that the energy does not increase we follow Levenberg and Marquardt [25] and add a term $\rho\|\boldsymbol{\Delta}\vartheta_i\|^2$ where the factor $\rho$ is multiplied by 10 whenever the update increases the energy, otherwise divided by 10. The resulting updates are given by

$$\begin{aligned} \boldsymbol{\Delta}\vartheta_i \;&= \mathbf{M}_i^{-1} \cdot \Big( \sum_{\hat{\mathbf{x}}, t} \chi_i(\mathbf{h}_i(\hat{\mathbf{x}}, t), t) \cdot \\ &\quad \big(I_i(\hat{\mathbf{x}}) - I(\mathbf{h}_i(\hat{\mathbf{x}}, t), t)\big) \mathbf{S}(\hat{\mathbf{x}}, t)^\top \nabla I(\mathbf{h}_i(\hat{\mathbf{x}}, t), t) \Big) \end{aligned}$$

with

$$\begin{aligned} \mathbf{M} &\leftarrow \rho\,\mathbf{I} + \sum_{\hat{\mathbf{x}}, t} \Big[ \chi_i(\mathbf{h}_i(\hat{\mathbf{x}}, t), t) \cdot \\ &\quad \mathbf{S}(\hat{\mathbf{x}}, t)^\top \nabla I(\mathbf{h}_i(\hat{\mathbf{x}}, t), t) \nabla I(\mathbf{h}_i(\hat{\mathbf{x}}, t), t)^\top \mathbf{S}(\hat{\mathbf{x}}, t) \Big] \end{aligned}$$

and where $\mathbf{I}$ is the identity matrix.

*2) Updating the Nonparametric Models:* Updating motion models when using super-resolution is a difficult problem and to our knowledge so far nobody managed to give gradient-based solutions. The only solution we know of is an exhaustive search over some putative motion models [15]. This breaks down if – as in our case – the motion models are high dimensional.

To update the nonparametric fields, we compute a proposal solution by warping the layer to each input frame according to the current motion model, then applying the method of Papenberg et al. [28]. The obtained velocity field may have higher cost than the previous one, in which case it is discarded.

### B. Update of the Layer Intensities

For optimizing the layer intensities, again we differentiate between the two functionals. In both cases the functional is convex with respect to the layer intensities, so gradient descent leads to the globally optimal layer appearance. Also, both functionals allow to estimate the intensities for each layer separately.

*1) Layer Intensities for the Basic Functional:* With respect to the layer intensities functional (6) is an instance of the ROF-model [30] - a point-wise quadratic data term in combination with a total variation regularity term. Numerous methods exist to minimize such functionals, including duality-based approaches [7], [43] which make use of Gauss-Seidel solvers or gradient descent.

In the special case where the total variation term is turned off ($\lambda = 0$) or replaced by e.g. area, the intensities are given as the average of the intensities along the trajectory of each point. Since we use the functional (6) only to initialize (10), we neglect the total variation term for the update of the layer intensities. Note that this term is still used for the optimization of the layer domains.

*2) Layer Intensities for Super-resolution:* In combination with super-resolution, the layers are estimated in a higher resolution than the input frames. We usually use a factor of 3 in each dimension, which increases the number of intensity variables by a factor of 9. The huge number of variables makes regularization terms like the total variation (3) essential: as long as there are 8 or less input frames, one has more variables to estimate than input data, so additional terms are needed to guarantee a single solution.

We smooth the absolutes in the data term, i.e. instead of $|x|$ we use the function $\Psi(x) = \sqrt{x^2 + \varepsilon}$, with $\varepsilon = 0.05$. The arising sub-problem of (10) is still convex in $I_i$, but now has a more complicated structure. Where previously Gauss-Seidel schemes were applicable, now we use gradient descent to obtain the global minimum. The functional derivative in the direction of an image $\eta : \mathbb{R}^2 \to \mathbb{R}$ is defined as

$$\left.\frac{\partial E}{\partial I_i}\right|_\eta = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big[ E(I_i + \epsilon\eta) - E(I_i) \Big] \quad .$$

This leads to gradient descent with the gradient

$$\begin{aligned} \frac{\partial E}{\partial I_i}(\hat{\mathbf{y}}) = &\Bigg[ \sum_{\mathbf{x}, t} \chi_i(\mathbf{x}, t)\, \chi_{A(\mathbf{x})}(\mathbf{h}_i(\hat{\mathbf{y}}, t)) * b(\mathbf{h}_i(\hat{\mathbf{y}}, t)) \\ &\cdot \Psi'\Big( \int_{A(\mathbf{x})} b(\mathbf{x}') * I_i(\mathbf{h}_i^{-1}(\mathbf{x}', t)) - I(\mathbf{x}, t) d\mathbf{x}' \Big) \Bigg] \left| \frac{d\mathbf{h}_i(\hat{\mathbf{y}}, t)}{d\hat{\mathbf{y}}} \right| \\ &- \lambda \operatorname{div} \left( \frac{\nabla I_i(\hat{\mathbf{y}})}{|\nabla I_i(\hat{\mathbf{y}})|} \right) \quad . \end{aligned}$$

where $\chi_{A(\mathbf{x})}$ is the indicator function for the area of pixel $\mathbf{x}$. For the simpler case of squared differences, a detailed derivation is given in [31].

This evolution equation can be interpreted as follows: the first term drives the layer intensities (after blurring) towards the observed intensities, whereas the second leads to a nonlinear, discontinuity preserving diffusion. The practical implementation is close to the book chapter [44]. For details see also the first author's thesis [31].

In practice gradient descent is not the best choice to solve for the layer intensities. Instead we resort to a primal-dual algorithm [43] where one simultaneously performs a gradient descent on the original (primal) functional and its dual formulation. The latter affects only the total variation term. This algorithm results in a speed-up of at least an order of magnitude.

To deal with the large number of intensity variables we resort to a GPU-based implementation and perform $250$ iterations of the primal dual algorithm. Each of the layers in Figure 7 is now estimated in $25$ seconds on a GTX 280 graphics card. This is for $30$ input frames of size $350 \times 240$. Layers are estimated in triple super-resolution.

*3) Estimating Intensities Outside the Layer Domains:* The procedure discussed so far only gives layer intensities inside the current layer domains. Yet, for the update of layer domains we need the layer intensities to be defined everywhere – the decision whether or not a layer is defined in a certain place depends upon whether the intensity at this place is in accordance with the input video.

To define the intensities outside the layer domain, we relax the formulation a bit. To be precise, we relax the visibility term (8) such that each layer is partially visible in each video position. To this end, the else-case in (8) is modified from $0$ to a small positive $\epsilon$. In the case of the basic cost functional (6) (where the data term is rewritten as in (10)) we are able to compute the limit of $\epsilon \to 0$: the intensities inside the domains are as before inside the domains, outside they are given as the average over all video points along the respective motion trajectory.

For the super-resolution cost (10) we use $\epsilon = 0.01$ and apply gradient descent. In practice this choice is small enough not to influence the intensities *inside* the domains.

### C. Update of the Layer Domains

It remains to solve for the layer domains $\Omega_i$. This procedure is virtually the same for both energies (6) and (10), so we exemplarily detail it for the basic cost (6).

We cast the problem as a binary *labeling problem* where the variables denote the characteristic functions $l_i(\cdot)$ of the layer domains. Solving this is intricate since the layer domains affect four terms in the functional:

- the **data term**, since the layer domains determine which layer is visible at each video position. This is actually a complex dependence: all $N$ layers have to be considered to determine the visibility at a given video pixel.
- the **boundary length**, which obviously depends on the layer domains.
- the **total variation term** since the domain of the each integral is the respective layer domain. This is one of the key factors to remove the mentioned trivial global optima.
- the **constraint** (2) demands that for each video pixel there is *some* associated layer.

This last point – to ensure the constraint – is resolved by minimizing the functional

$$E(\{\Omega_i, I_i, \mathbf{h}_i\}) \;+\; \gamma \sum_{\mathbf{x},t} \Big(1 - \sum_i \chi_i(\mathbf{x},t)\Big) \quad , \qquad (12)$$

where $\gamma$ is set to the last determined energy.

The second major difficulty, caused by the data term, is not so easy to handle: since visibility depends on all $N$ layers at once, we get a term of order $N$ (also called $N$-ary) in the objective labeling function. For the case of 2 layers we give a globally optimal solution, based on graph cuts. For the case of $N > 2$ layers such a solution generally does not exist: for $N = 3$ layers the (ternary) data term for a single pixel does not satisfy the submodularity conditions in [14], [2], [21] and this property remains when flipping the roles of $0$ and $1$ for any number of variables[2].

*1) Graph Cut-based Optimization:* The arising labeling problem is formulated on the layer domains. To this end, for each layer $i$ a discretized a set of spatial positions that could potentially belong to the layer is set up (from the current motion models) and denoted $D_i$. The task is now to determine a binary variable $l_i(\hat{\mathbf{x}})$ for each $\hat{\mathbf{x}} \in D_i$ and each layer $i$.

With the mentioned representation it is easy to include the notions of length and region integrals into the graph. The length of each layer boundary is approximated as in [5] – each layer pixel $\hat{\mathbf{x}}$ is connected to a set of neighbors (we take the set $\mathcal{N}_8(\hat{\mathbf{x}})$ of the $8$ closest neighbors) via an edge with a suitable edge weight:

$$|\partial\Omega_i| \;\approx\; \sum_{\hat{\mathbf{x}} \in D_i} \sum_{\hat{\mathbf{y}} \in \mathcal{N}_8(\hat{\mathbf{x}})} \frac{\nu}{\|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|} (1 - \delta(l_i(\hat{\mathbf{x}}), l_i(\hat{\mathbf{y}}))) \quad ,$$

where $\delta(\cdot,\cdot)$ is the Kronecker-$\delta$. The region integrals are approximated by a sum of unary terms (at the cost of a slight imprecision along the region boundary):

$$\int_{\Omega_i} |\nabla I_i(\hat{\mathbf{x}})| \, d\hat{\mathbf{x}} \;\approx\; \sum_{\hat{\mathbf{x}} \in D_i} l_i(\hat{\mathbf{x}}) \, |\nabla I_i(\hat{\mathbf{x}})| \quad .$$

Both the region and the length terms are submodular and hence easy to minimize. A key factor in the following will be that these terms remain submodular when the roles of $0$ and $1$ are flipped for an *entire* labeling function $l_i$ (here the formulation of the region term needs to be adapted).

The remaining terms – the data terms (including visibility reasoning) and the constraint – are sums over video pixels, not layer variables. For the implementation they are grouped together, resulting in one $N$-ary term per video pixel $(\mathbf{x}, t)$. This term depends on one variable in each layer. The corresponding spatial position in layer $i$ is denoted $\hat{\mathbf{x}}_i = \mathbf{h}_i^{-1}(\mathbf{x}, t)$ for given $(\mathbf{x}, t)$. Here we round to the nearest pixel position to get the corresponding layer variable, but use subpixel interpolation to

[2]This implies that for a video consisting of a single pixel the arising function cannot be optimized with graph cuts. It is unlikely that for larger videos the functional is submodular when it contains terms that are not.

determine the layer intensity. The terms are then written as

$$\sum_{\mathbf{x},t} \Big[ \sum_i \chi_i(\mathbf{x},t \,|\, \{\Omega_i\},\{\mathbf{h}_i\}) \left(I(\mathbf{x},t) - I_i(\hat{\mathbf{x}}_i)\right)^2 + \gamma\Big(1 - \sum_i \chi_i(\mathbf{x},t \,|\, \{\Omega_i\},\{\mathbf{h}_i\})\Big)\Big] \tag{13}$$

where we have explicitly indicated the dependence of the visibility $\chi_i(\cdot,\cdot)$ on the layer domains: this dependence makes optimization difficult. In the implementation we differentiate between the two layer case and the multi layer case.

*2) The Two Layer Case:* For two layers each term in (13) can be written as a binary submodular term depending on the variables $l_1(\hat{\mathbf{x}}_1)$ and $l_2(\hat{\mathbf{x}}_2)$ for the respective $(\mathbf{x},t)$. The constraint term simply serves to prevent that both variables are labeled as 0, which would leave the visibility undefined. This constellation is consequently penalized with $\gamma$. For the remaining three constellations one chooses the intensity difference to $I_1(\hat{\mathbf{x}}_1)$ if $l(\hat{\mathbf{x}}_1)=1$, otherwise the difference to $I_2(\hat{\mathbf{x}}_2)$:

| | |
|---|---|
| $E^{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(0,0)$ | $\gamma$ |
| $E^{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(0,1)$ | $\big[I(\mathbf{x},t) - I_2(\hat{\mathbf{x}}_2)\big]^2$ |
| $E^{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(1,0)$ | $\big[I(\mathbf{x},t) - I_1(\hat{\mathbf{x}}_1)\big]^2$ |
| $E^{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(1,1)$ | $\big[I(\mathbf{x},t) - I_1(\hat{\mathbf{x}}_1)\big]^2$ |

In this form the term is not submodular, but it can be made so by invert the labeling $l_2(\cdot)$. The function (12) is now written as a sum of submodular, binary terms and can hence be optimized globally via graph cuts [14], [21].

*3) The Multi Layer Case:* In the case of $N > 2$ layers, the function (12) becomes a non-submodular function of binary variables which contains terms of order $N$.

The key idea of our approach is to exploit the related *multi*-label problem given by the labeling (1). Multi-label problems are often addressed by expansion moves [6]. We adopt this approach to our *binary* labeling problem and, starting from a consistent labeling, design one move for each layer $i$. Each move is a binary submodular labeling problem containing unary and binary terms. Note that since problem (12) is not of metric form, we cannot give the usual factor guarantee for expansion moves [6].

In the move for layer $i$ all variables $\hat{\mathbf{x}} \in D_i$ in layer $i$ are allowed to *join* the layer, i.e. $l_i(\hat{\mathbf{x}})$ may either be set to 1 or kept at its previous value. Simultaneously all variables $\hat{\mathbf{y}} \in D_j$ for all layers $j \neq i$ are allowed to *leave* the support of layer $j$. That is, $l_j(\hat{\mathbf{y}})$ may either be set to 0 or kept at its previous value. It is important to consider all layers at once: otherwise, once a layer $j$ becomes visible at a video position, it could never be replaced by a layer $i > j$.

For each video position $(\mathbf{x},t)$ there are two possibilities in the move for layer $i$: either layer $i$ becomes visible or the previously visible layer, say $j$, remains visible. This decision is reflected by an additional variable $\tilde{l}(\mathbf{x},t)$. These variables allow to represent the data terms as unary terms, where a value of 1 signals that $i$ becomes visible, 0 that $j$ remains visible.

It remains to ensure that the variables $\tilde{l}(\mathbf{x},t)$ reflect the visibility as induced by the layer labelings $l_1,\dots,l_N$. This means that three constraints have to be ensured:

1) If $i$ becomes visible ($\tilde{l}(\mathbf{x},t) = 1$), then the respective layer point $\hat{\mathbf{x}}_i$ must be in the support of layer $i$, i.e. $l_i(\hat{\mathbf{x}}_i)$ must be 1.
2) If $j$ remains visible ($\tilde{l}(\mathbf{x},t) = 0$) and $i < j$, then $\hat{\mathbf{x}}_i$ may not be in the support of $i$, i.e. $l_i(\hat{\mathbf{x}}_i)$ must be 0. Otherwise the occlusion order would be violated.
3) If $i$ becomes visible ($\tilde{l}(\mathbf{x},t) = 1$) and $i > j$, then for all $j \leq k < i$ the variable $l_k(\hat{\mathbf{x}}_k)$ must be set to 0, again to respect the occlusion order.

Each constraint can be written as (sums of) binary terms combining a video variable with a layer variable, with the cost of 0 for one constellation and $\gamma$ for all others. The terms are submodular if we invert all labelings for layers $j \neq i$.

*4) Alternative:* In principle, much of the cost can also be optimized in the video space. The total variation term would then be encoded as a concave potential with one breakpoint [20]. It turns out that the introduced auxiliary variable corresponds to a layer variable in our setting. These variables are needed explicitly to encode the boundary regularization in the layer space. They are also very helpful for a simple and memory-saving implementation of enforcing the layer order consistency.

### D. Optimizing the Layer Order

The presented occlusion model depends heavily on which layer is deemed to have the number 1, 2 and so forth. In practice it is sensible to optimize this order rather than fix it beforehand. Yet, we know of no better solution than a brute-force search over all $N!$ permutations. For two layers this search is included in the alternating minimization scheme: The multi-scale scheme is run for both orders up to the original scale. Then the order with lower energy is chosen. We also tried to optimize the order at each individual level. Yet, this leads to wrong decisions early on which are never corrected in later stages.

For more than 2 layers the computational cost become too high since optimization is already based on iterating graph cuts. Instead we assume that faster moving layers correspond to lower occlusion orders, i.e. we update the order after every iteration. This assumption includes, but is not limited to, all cases with a static scene and a moving camera.

## VI. Experiments

The presented contributions and improvements will now be evaluated on four real-world sequences containing up to 30 frames each.

The stated coding cost include three parameters: $\lambda, \nu$ and the width of the blurring kernel. We found $\lambda = 275\,T$ and $\nu = 0.33\,T$ to give consistently good results[3], where $T$ is the number of input frames. Yet, in the end the choice is heuristic. We have therefore indicated the one case where we differed from these parameters.

The width of the blurring kernel was adjusted manually for each sequence. For off-the-shelf usage we recommend a width

[3]These parameters are for the basic coding cost. For the refined cost they are divided by 50 to account for the switch from squared to absolute differences in the data term.

of $\sigma = 0.5\,L$, where $L$ is the super-resolution factor in each dimension.

A first experiment is given in Figure 5 for the coastguard sequence. This sequence is difficult because of the floating water which does not agree with the model assumptions. Still, the boat as well as the correct layer order are identified.

### A. Super-resolved Layer Decompositions

We start by giving results for the refined coding cost functional. We tested three different sequences, each with its own difficulty: for the Pickup Sequence[4] in Figure 3[5], one deals with specularities on the can. In the Avengers Sequence shown in Figure 6 the background is moving faster than the foreground. In both cases the algorithm determines the correct layer order.

The last sequence is the Flower Garden Sequence, where input images and resulting layers are shown in Figures 1 and 7. Here some layers contain objects with mixed depths, which requires a non-parametric motion model. The sequence also demonstrates that more than two layers can be handled: we initialized with $4$ layers, one of them vanished during optimization.

Both the Pickup Sequence and the Flower Garden Sequence are long sequences with 31 and 30 frames respectively. Still it is possible to decompose them into just 2 or 3 layer images. Also, in all cases very precise motion boundaries were found. This is due to the physically consistent occlusion model.

### B. Robustness of Parameters

Since the blurring kernel is set by the user, the proposed cost function involves effectively two free parameters, $\lambda$ and $\nu$. While above we have given a setting that works well for all considered sequences, we now also evaluate how changing the parameters affects the results.

Figure 8 evaluates the effect of changes in $\lambda$ and $\nu$ on the Avengers Sequence. Here it can be seen that small $\lambda$s result in speckled regions, but that otherwise the value is not very critical. For $\nu$ the results are quite similar within a range of a factor of $100$. Large values result in only one layer (but note that it is not the unrolled image sequence).

### C. Basic Cost, Refined Cost and Robust Data Terms

In this paper, we have proposed two different cost functionals. As Figure 9 shows, the basic cost suffices to get tight region boundaries with consistent occlusion reasoning.

Yet, if one also wants a precise reconstruction of the scene, the refined cost become crucial: by modeling physical details of the image formation process, one can infer very small details that are not visible in any of the input frames.

When introducing the refined cost, we argued that the absolute differences in the data term robustify the layer estimation process. Indeed, it is well known throughout the literature that absolute differences are robust to outliers, whereas for squared ones already a single outlier can lead to arbitrarily bad results. Yet, such effects are usually demonstrated on synthetic data, e.g. by adding salt-and-pepper noise.

In this paper we justify the robust terms on real-world data. Figure 10 shows two reconstructions for the Pickup Sequence. Since in one frame a specularity on the metal can is visible, the squared differences produce artifacts. In this case they might still be acceptable.

This is different when using the nonparametric motion models: as shown in Figure 11 the squared data terms produce severe artifacts when the velocity estimates are imperfect. Once present, these effects cannot be compensated by iterating the processes. To be consistent with the model, for the squared differences we used the method of Horn and Schunck to compute proposal velocity fields (this also changes the terms $R(\mathbf{h}_i)$ which now takes the squared gradient absolute). We checked that the same proposals do not cause artifacts when using absolute differences.

### D. Parametric vs. Nonparametric Motion

Above we have already shown that the use of nonparametric velocities results in fine-detailed layer images for the Flower Garden Sequence. Figure 12 demonstrates the influence of these motion models: with the parametric part alone, many parts remain blurry. The reason is that the layer contains objects of different depths. E.g. the trees actually stand in front of the houses. With the nonparametric model it is still possible to obtain a single, sharp layer image.

### E. Comparison to Alternative Approaches

Finally, we compare our method to other approaches on motion analysis: In Figure 13 we show results for the layer decomposition approach of Kumar et al. [23] and two approaches to motion segmentation: the space-time motion segmentation of Cremers and Soatto [9] and an extension of the graph cut motion segmentation [32] which includes image warping.

These results demonstrate that near occlusions motion segmentation does not provide tight region boundaries. In particular, none of the two methods identifies the region between the fingers.

This is different for layer decomposition. While our method can handle the entire 31 frames, the method of Kumar et al. suffers from significant drift [22] when run on more than $10$ frames. Moreover, it splits the thumb into two parts. For fairness it must be noticed that this method was designed for articulated motion.

## VII. Discussion

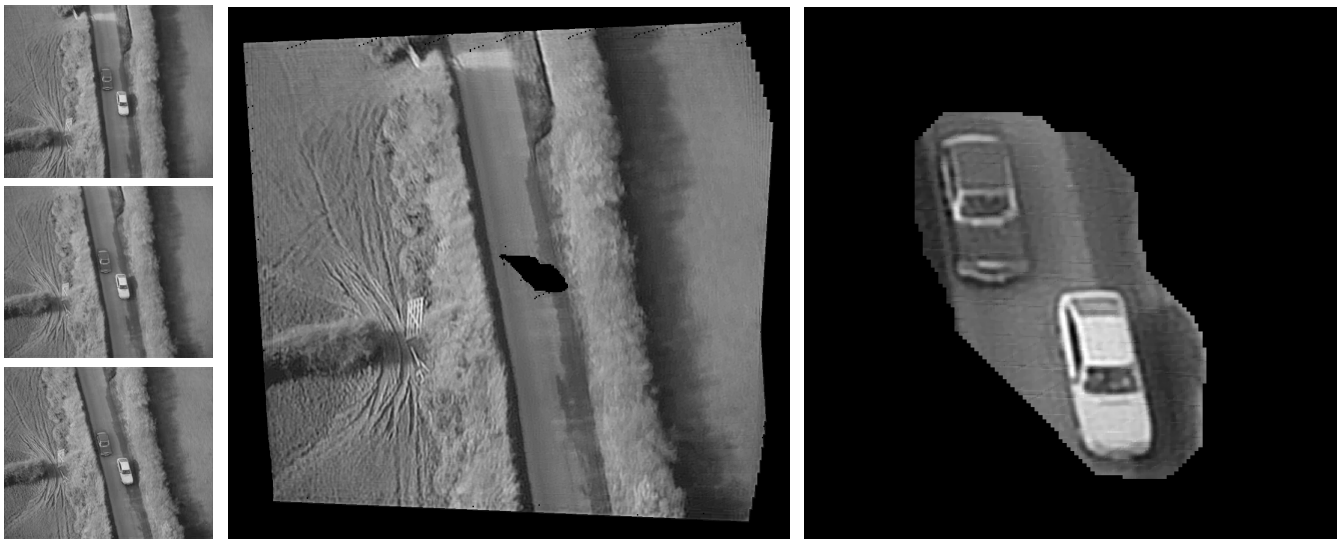
While this paper has made a number of interesting contributions, the problem of layer decomposition as a whole is certainly not solved yet. We now discuss a number of issues:

**Robustness of Parameters.** We have given a parameter setting that works well on all tested sequences. In addition, it was shown above that the algorithm is fairly robust against the

[4]Image data courtesy of Michael Black, http://www.cs.brown.edu/~black/ . We use frames $90 - 120$. The Flower Garden Sequence is obtained from the same source.

[5]To get optimal super-resolved layers, we use $\nu = 500\,T$. The standard settings produce nearly the same layer shapes, but more noise in the layer images.

Fig. 5. Induced video regions for frame 1, 15 and 30 of the coastguard sequence.



3/9 input images super-resolved background layer super-resolved foreground layer

Fig. 6. Input frames and obtained super-resolved layers for the Avengers sequence. The algorithm correctly determines that the background is moving faster than the foreground.

3/30 input images super-resolution layer 1 super-resolution layers 2 & 3

Fig. 7. Layers and induced region boundaries obtained for the Flower Garden Sequence (three of the 30 input frames are shown in Figure 1): the combination of the absolute differences in the data term and the nonparametric velocity fields allows to get a fine-detailed notion of the scene.

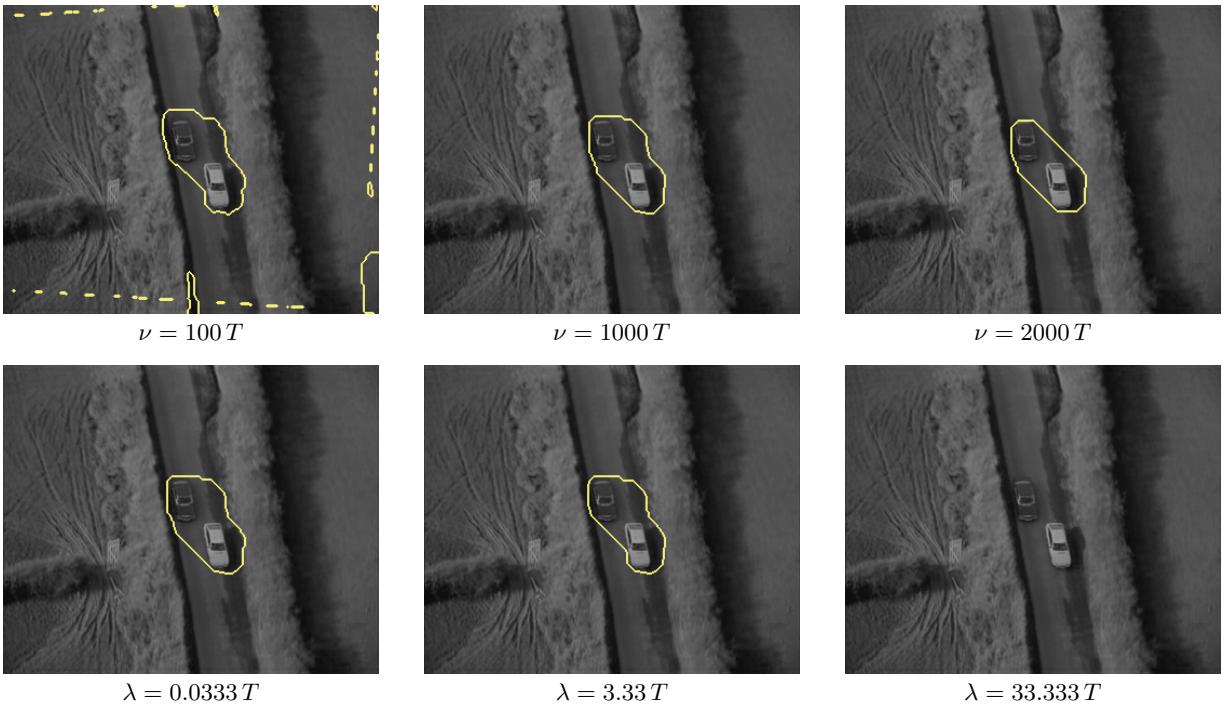



Fig. 8. The choice of parameters affects the results only mildly.


Fig. 9. The basic cost provides tight region boundaries, the refined cost adds precise layer images.

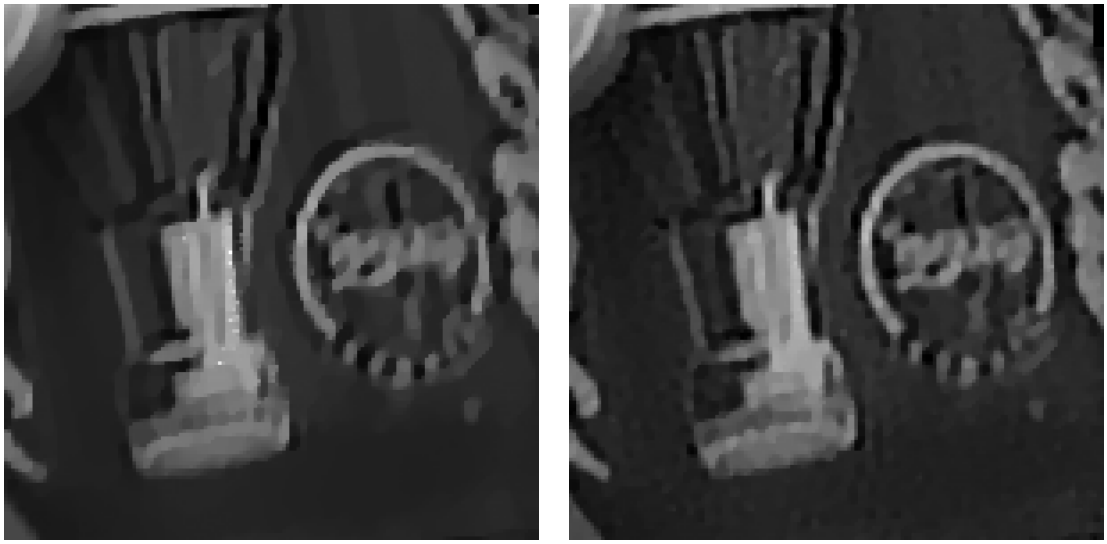



Fig. 10. Effect of robust and non-robust data terms, demonstrated on real-world data: robust terms improve the results in the presence of lighting changes.

with squared differences, incorrect velocities cause severe artifacts.

absolute differences are robust against erroneous velocities.

Fig. 11. In combination with nonparametric motion models, the robust data term proves crucial to get acceptable layer images.

with parametric motion: blurry where objects stand out.

with non-parametric motion: sharp everywhere.

Fig. 12. Non-parametric motion models are needed when there are objects with different depths in the same layer.

choice of parameters. In fact, it is well known that problems tend to become more robust when several frames of a video sequence are considered at once. Still, there is the issue of local minima that is present in virtually all approaches to motion estimation (with reasonably large displacements).

**Adjusting the blurring kernel.** Currently the variances of the blurring kernel are set by hand. In practice this is not much of a hindrance since it has to be repeated only if the camera or the focal length is changed. Moreover, we have given a parameter for off-the-shelf usage. Strategies to automatically derive this parameter have been considered, e.g. [41], but are outside the scope of this work.

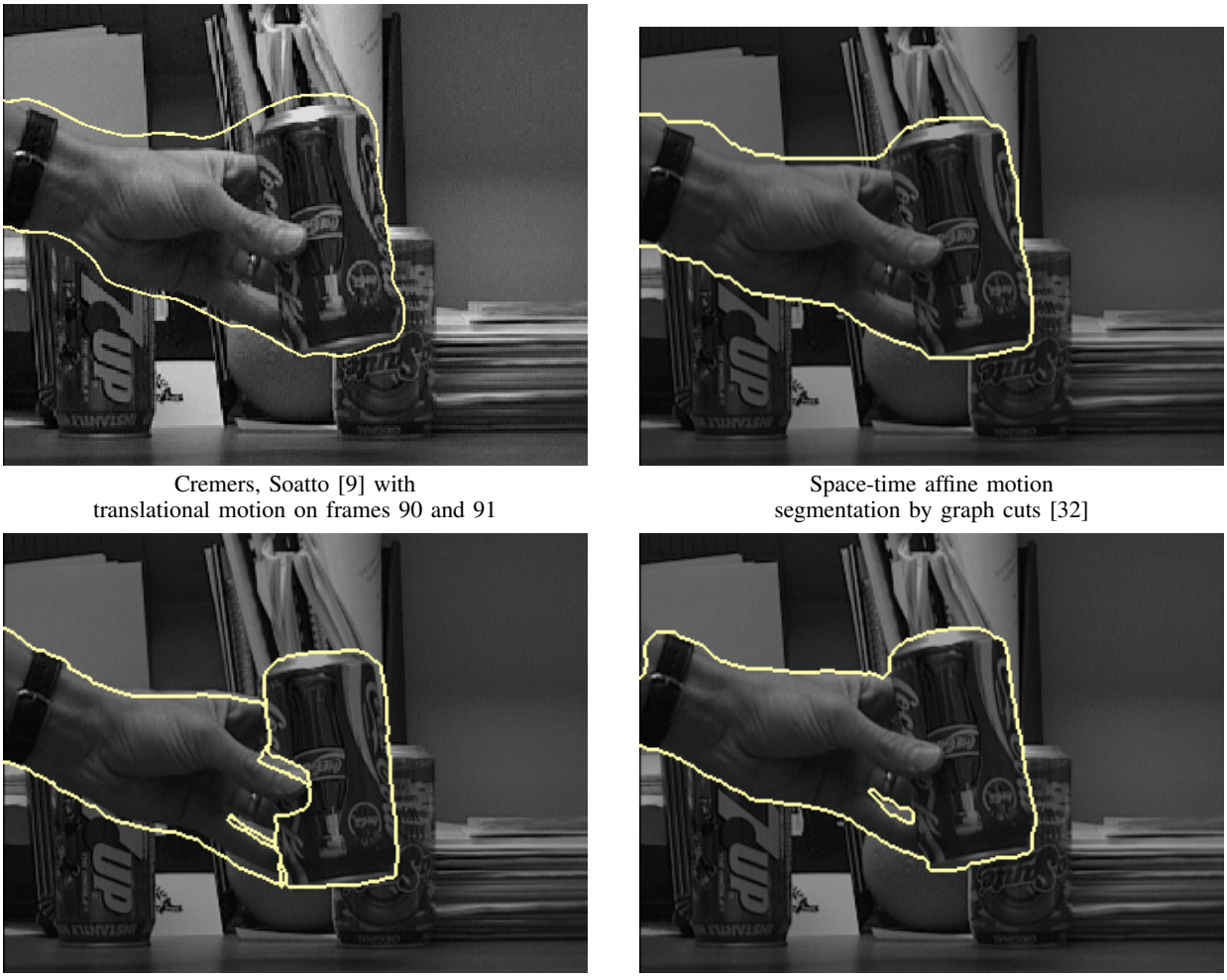
**Choosing the number of layers.** At present the number of layers is considered given, and our attempts to automatically select it were discouraging. We note however, that there is a large body of literature devoted to this subject, and we do not mean to claim that our work renders these approaches obsolete. On the contrary, we consider these topics to be orthogonal and are confident that the ideas can be combined.

**Initialization.** We have chosen a simple initialization to demonstrate the power of our contributions, which are based on a rigorous energy minimization framework. More refined initializations can of course easily be included, and again we refer to the large body of literature available.

**Length Regularity.** Virtually all works combining motion estimation and shape optimization that include a regularity term make use of some kind of length regularity. It is well known that such terms have difficulties to recover long and elongated structures. Recently, researchers have considered curvature regularity to remove this shrinking bias [34]. The inclusion of such terms is left for future work.

## VIII. Conclusion

We have introduced a layer decomposition approach based on minimizing coding cost functionals. Compared to other approaches to motion analysis – motion estimation, motion segmentation and layered motion segmentation – layer decomposition is based on a fully generative model and therefore



Cremers, Soatto [9] with translational motion on frames 90 and 91

Space-time affine motion segmentation by graph cuts [32]

Kumar et al. [23] run on frames 90–99

Region boundaries induced by the proposed layer partitioning

Fig. 13. Comparison on the Pickup Sequence: with the proposed method, the tightest boundaries are obtained.

incorporates a natural treatment of occlusion. Moreover, the proposed layer formulation is based on a physically more realistic model of the image formation process (including lens blur and downsampling). As a result of this, we obtain a more detailed scene structure than any single input frame would allow.

We have presented a number of contributions concerning the model as well as the optimization procedure. For the model, we have first shown that the formulation by Jackson et al. removes trivial optima that arise with previous formulations. Further, an image formation model which is more realistic than that of previous layer approaches – including lens blurring and downsampling – allows for layer images of a resolution superior to that of the individual input images.

Optimization of the coding cost functional is done by alternating the estimation of geometry, intensity and motion of all layers. The geometry estimation integrates spatial smoothness and region-based priors. It is solved by means of graph cuts in the layer domain (rather than the input domain). The optimization of layer intensities gives rise to a convex total variation deblurring, providing crisp high-resolution layer images. And the motion estimation is solved by either high-order parametric (in space and time) motion models, or by state-of-the-art nonparametric variational optic flow techniques.

In numerous experiments, we demonstrated that a given video can be decomposed into a superposition of super-resolved moving layers by minimizing the proposed coding cost. In general this is a very large computational effort, where most of the runtime is spent on estimating the layer intensities. GPU implementations lead to acceptable runtimes.

## ACKNOWLEDGMENTS

We thank Frank R. Schmidt, Thomas Brox and Thomas Pock for helpful discussions. This research was supported by the German Research Foundation, grants #CR-250/1-1 and #CR-250/2-1 and the ERC Starting Grant “ConvexVision”.

**Thomas Schoenemann** received a BS (Vordiplom, 2002) and the MS (Diplom, 2005) in Computer Science from the Technical University RWTH Aachen, Germany. In 2009 he obtained a Ph.D. in the Computer Vision Group at the University of Bonn, Germany. Subsequently he joined the Vision Group at Lund University, Sweden. Since 2011, he is with the Computer Science Department at the University of Pisa, Italy. He is generally interested in applied optimization and artificial intelligence.

**Daniel Cremers** obtained a PhD in Computer Science from the University of Mannheim, Germany. Subsequently he spent two years as a postdoctoral researcher at the University of California at Los Angeles and one year as a permanent researcher at Siemens Corporate Research in Princeton, NJ. From 2005 to 2009 he headed the Research Group for Computer Vision, Image Processing and Pattern Recognition at the University of Bonn, Germany. Since 2010, he holds the chair for Computer Vision and Pattern Recognition at TU Munich, Germany. His research is focused on optimization methods in computer vision. He received several awards, including the *Best Paper of the Year 2003* by the Pattern Recognition Society, the *Olympus Award 2004*, and the *UCLA Chancellor's Award for Postdoctoral Research 2005*.

## REFERENCES


[1] S. Ayer and H.S. Sawhney. Layered representation of motion video using robust maximum likelihood estimation of mixture models and MDL encoding. In *IEEE International Conference on Computer Vision (ICCV)*, Boston, Massachusetts, June 1995.

[2] A. Billionet and M. Minoux. Maximizing a supermodular pseudo-boolean function: A polynomial algorithm for supermodular cubic functions. *Discrete Applied Mathematics*, 12(1):1–11, 1985.

[3] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *IEEE International Conference on Computer Vision (ICCV)*, Corfu, Greece, September 1999.

[4] M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise smooth flow fields. *Computer Vision, Graphics and Image Processing: Image Understanding*, 63(1):75–104, 1996.

[5] Y. Boykov and V.N. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *IEEE International Conference on Computer Vision (ICCV)*, Nice, France, October 2003.

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239, 2001.

[7] A. Chambolle. An algorithm for Total Variation minimization and applications. *Journal on Mathematical Imaging and Vision*, 20(1-2):89–97, January 2004.

[8] D. Cremers and S. Soatto. Variational space-time motion segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, Nice, France, October 2003.

[9] D. Cremers and S. Soatto. Motion competition: A variational framework for piecewise parametric motion segmentation. *International Journal on Computer Vision (IJCV)*, 62(3):249–265, 2005.

[10] R. Dupont, O. Juan, and R. Keriven. Robust segmentation of hidden layers in video sequences. In *International Conference on Pattern Recognition (ICPR)*, Hong Kong, China, September 2006.

[11] R. Dupont, N. Paragios, R. Keriven, and P. Fuchs. Extraction of hidden layers of similar motion through combinatorial techniques. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, St. Augustine, Florida, November 2005.

[12] S. Farsiu, M.D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing (TIP)*, 13(10):1327–1344, 2004.

[13] B.J. Frey, N. Jojic, and A. Kannan. Learning appearance and transparency manifolds of occluded objects in layers. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, Wisconsin, June 2003.

[14] P.L. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13:388–399, 1965.

[15] R.C. Hardie, K.J. Barnardt, and E.E. Armstrong. Joint MAP registration and high resolution image estimation using a sequence of undersampled images. *IEEE Transactions on Image Processing (TIP)*, 6(12):1621–1633, December 1997.

[16] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[17] J.D. Jackson, A.J. Yezzi, and S. Soatto. Dynamic shape and appearance modeling via moving and deforming layers. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, St. Augustine, Florida, November 2005.

[18] J.D. Jackson, A.J. Yezzi, and S. Soatto. Dynamic shape and appearance modeling via moving and deforming layers. *International Journal on Computer Vision (IJCV)*, 79(1):71–84, August 2008.

[19] N. Jojic and B. Frey. Learning flexible sprites in video layers. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Kauai Marriott, Hawaii, June 2001.

[20] P. Kohli, L. Ladickỳ, and P.H.S. Torr. Robust higher order potentials for enforcing label consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 82(3):302–324, 2009.

[21] V.N. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(5):657–673, 2004.

[22] M. Pawan Kumar and P.H.S. Torr. Personal correspondence, 2007.

[23] M. Pawan Kumar, P.H.S. Torr, and A. Zisserman. Learning layered motion segmentations of video. *International Journal on Computer Vision (IJCV)*, 76(3):301–319, March 2008.

[24] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, Canada, 1981.

[25] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal of Applied Mathematics*, 11:431–441, 1963.

[26] E. Memin and P. Perez. Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Transactions on Image Processing (TIP)*, 7(5):703–719, 1998.

[27] J.-M. Odobez and P. Bouthemy. Direct incremental model-based image motion segmentation for video analysis. *Signal Processing*, 66:143–155, 1998.

[28] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal on Computer Vision (IJCV)*, 67(2):141–158, April 2006.

[29] A. Rav-Acha, P. Kohli, C. Rother, and A.W. Fitzgibbon. Unwrap mosaics: A new representation for video editing. In *ACM SIGGRAPH*, August 2008.

[30] L.I. Rudin, S.J. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[31] T. Schoenemann. *Combinatorial solutions for shape optimization in computer vision*. PhD thesis, Universität Bonn, Bonn, Germany, April 2009.

[32] T. Schoenemann and D. Cremers. Near real-time motion segmentation using graph cuts. In *Pattern Recognition (Proc. DAGM)*, Berlin, Germany, September 2006.

[33] T. Schoenemann and D. Cremers. High resolution motion layer decomposition using dual-space graph cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, June 2008.

[34] T. Schoenemann, F. Kahl, S. Masnou, and D. Cremers. A linear framework for region-based image segmentation and inpainting involving curvature penalization. *International Journal on Computer Vision (IJCV)*, 2012. To appear.

[35] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, Washington, June 1994.

[36] E.P. Simoncelli. *Distributed representation and analysis of visual motion*. PhD thesis, Massachusetts Institute of Technology (MIT), Dept. of Elect. Eng. and Comp. Sci., Cambridge, Massachusetts, 1993.

[37] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing (TIP)*, 3(5):625–638, 1994.

[38] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *IEEE International Conference on Computer Vision (ICCV)*, Kyoto, Japan, September 2009.

[39] C. Williams and M. Titsias. Greedy learning of multiple objects in images using robust statistics and factorial learning. *Neural Computation*, 16(5):1039–1062, 2004.

[40] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1644–1659, 2005.

[41] Y. You and M. Kaveh. A regularization approach to joint blur identification and image restoration. *IEEE Transactions on Image Processing (TIP)*, 5(3):416–428, 1996.

[42] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Pattern Recognition (Proc. DAGM)*, Heidelberg, Germany, September 2007.

[43] M. Zhu and T. Chan. An efficient primal-dual hybrid algorithm for total variation image restoration. Technical report, UCLA, 2008.

[44] A. Zomet and S. Peleg. Super-resolution from multiple images having arbitrary mutual motion. In *Super-Resolution Imaging*. Kluver Academic, September 2001.