Hierarchically-Constrained Optical Flow

Ryan Kennedy and Camillo J. Taylor
Department of Computer and Information Science
University of Pennsylvania
{kenry,cjtaylor}@cis.upenn.edu

Abstract

This paper presents a novel approach to solving optical flow problems using a discrete, tree-structured MRF derived from a hierarchical segmentation of the image. Our method can be used to find globally-optimal matching solutions even for problems involving very large motions. Experiments demonstrate that our approach is competitive on the MPI-Sintel dataset and that it can significantly outperform existing methods on problems involving large motions.

1. Introduction

Optical flow algorithms attempt to estimate the perceived motion of each pixel between two successive frames of an image sequence [30]. Often, a grid-based graphical model is employed that connects each pixel with its immediate neighbors in the image and each variable can be considered as either continuous or discrete. While continuous-valued optimization allows for sub-pixel motion estimates [4], it is also subject to getting stuck in local optima. This is especially problematic for large motions, which are often missed in a local search.

In contrast, discrete graphical models can sometimes be solved optimally [23]. Even when the problems are NP-hard, approximate methods can often provide a bound on the globally-optimal energy [18]. These discrete optimization methods, however, tend to have computational complexities that are highly dependent on the size of the label space, and they cannot be easily used for complex optical flow problems that may involve offsets of several hundred pixels between frames [6].

In this paper, we propose the use of a tree-based graphical model derived from a hierarchical image segmentation. In any choice of model, there is an inherent tradeoff between the model's representational power and its optimization complexity. We argue that a hierarchical approach is capable of accurately modeling natural images while also being computationally tractable. Indeed, we show how the global optimum of this discrete optimization problem can be found by using efficient methods borrowed from the literature on deformable-parts models [9], even for problems involving hundreds of thousands of labels. This allows us to optimally solve large optical flow problems using a discrete, global model for the first time. We also introduce several small approximations that dramatically speed up the algorithm while still yielding solutions that are very-nearly optimal.

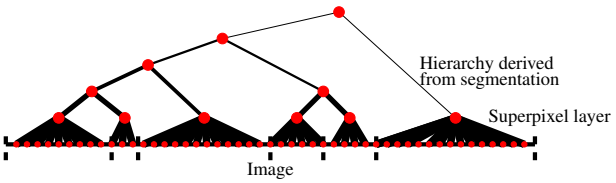


Figure 1: Depiction of our model, shown in 1D for simplicity. We use a hierarchical image segmentation, and each segment has an associated variable that is connected to its children. The root of the tree represents the entire image and leaves represent pixels. Edge weights are a function of the weights in the segmentation, denoted here by the thickness of black edges. Variables in the graphical model are denoted by red circles. After optimization, the final motion estimate is given by labels assigned to the pixel variables.

Additionally, we describe a simple method of incorporating information from multiple frames in optical flow. We use the idea of *inertial estimates* from [16], where several estimates of the optical flow are computed using nearby frames and subsequently fused using a classifier. We show how the inertial estimates can instead be directly modeled in our cost function.

In Section 6, we show that state-of-the-art motion estimation schemes based on local optimization can have difficulties even on relatively simple motion analysis problems that contain large displacements. We illustrate this issue with both synthetic datasets and real images and show how our proposed global method can significantly improve performance in these situations. We also evaluate the proposed method on the challenging MPI-Sintel dataset [6] and compare its performance to other recent methods.

2. Related Work

Discrete optimization is commonplace in stereo correspondence [19], but has not often been used in optical flow because of the large label spaces involved. In [15], it was shown that some image labeling problems can be reduced in size, after which exact optimization methods can be used, but this scheme is still only applicable to small label spaces. In [14], it was shown how multi-label problems can be solved if the label set has a linear ordering and the regularizer is convex. This was extended to continuous label spaces in [25], but it is restricted to scalar-valued labels. In [29] and [13], further relaxations were proposed that allow for near-optimal solutions to be found for some optical flow problems. However, their complexity is still strongly tied to the size of the label space, limiting their applicability.

Other related hierarchical methods for image matching include [33, 20, 17], although these methods are still approximate. Hierarchical methods have also been used in image labeling problems other than correspondence, including [11, 27, 34], where tree-based models were used for exact inference. All of these previous methods involve label spaces that are orders of magnitude smaller than those used in this paper.

3. Problem setup

Let $I_1, I_2 : (\Omega \subseteq \mathbb{R}^2) \to \mathbb{R}^d$ be two d-dimensional images, where d is typically 3 for color images and 1 for grayscale images. The image domain is denoted by Ω, which we consider to be a discrete set of pixel locations.

Our algorithm proceeds by constructing a tree-structured Markov random field (MRF) model based on a hierarchical segmentation of the image I_1 as depicted in Figure 1. The goal of the motion estimation procedure is to assign an integral offset to each vertex that maps it onto its correspondent in I_2. This is done by defining a cost function that is small when pixels have a good correspondence and when each node has a similar offset to its parent. The goal of the optimization procedure is to find a solution which minimizes this cost function. The elements of this procedure are described in more detail in the following subsections.

3.1. Hierarchical segmentation

We base our hierarchical segmentation of I_1 on the framework of [7], which is in turn based on the ultrametric contour map (UCM) hierarchy of [2, 3]. Rather than using a watershed segmentation as the the base of the hierarchy, we begin with a segmentation using SLIC superpixels [1]. By using SLIC, we are able to explicitly control the size and complexity of the base superpixel layer. In our experiments, we set the region size and regularization parameters to 50, which we found resulted in superpixels that are small enough to contain primarily a single motion but are large enough for an efficient algorithm.

Examples of segmentations obtained with this procedure are shown in Figure 4. The segmentation divides the image into a set of small superpixels that are then successively merged with their most similar neighbors to form a tree structure. Distances in the hierarchy encode the similarity of regions and the likelihood that they belong to the same object [3].

3.2. Graphical model

We construct a MRF model which mirrors the structure of the tree produced by the hierarchical segmentation (Figure 1). Let the graph be denoted by $G = (\mathcal{V}, \mathcal{E})$, for vertex set $\mathcal{V}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The vertex set $\mathcal{V}$ corresponds to the nodes in the segmentation tree; the leaf nodes correspond to the individual pixels and the interior nodes correspond to merged regions produced by the hierarchical segmentation. Each vertex has an associated *location*, *area* and *similarity weight*.

The location of a vertex in I_1 is denoted by the function $\mathbf{x} : \mathcal{V} \to \Omega$. For vertices corresponding to pixels, this is defined simply as the location of the pixel. For internal nodes, we arbitrarily define it as the pixel nearest to the region's centroid. The area of each vertex is denoted by $\mathbf{a} : \mathcal{V} \to \mathbb{R}_+$. This area is 1 for the vertices corresponding to individual pixels, and for internal nodes the area is the sum of the areas of its children. The similarity weight of each interior vertex v is denoted by $\mathbf{s} : \mathcal{V} \to \mathbb{R}_+$, and is defined as the similarity of v's two children. Specifically, our chosen segmentation algorithm [7] specifies an ultrametric distance $\mathbf{d}(v)$, which is the level at which the regions corresponding to v's children are merged in the segmentation process. The similarity weight is then set using a logistic function:

$$\mathbf{s}(v) = 1/\left(1 + e^{\eta_1(\mathbf{d}(v)+\eta_0)}\right) . \quad (1)$$

Edges in the graph are placed between all nodes sharing a parent-child relationship in the image segmentation hierarchy:

$$\mathcal{E} = \{(v_p \in \mathcal{V}, v_c \in \mathcal{V}) \mid v_p \text{ is the parent of } v_c\}. \quad (2)$$

Each edge $(v_p, v_c) \in \mathcal{E}$ is assigned a weight that determines how similar the offsets of a node v_p and its child v_c should be, which will be used in the smoothness term of our cost function. This weight is denoted by $\mathbf{w} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$, and is defined as

$$\mathbf{w}(v_p, v_c) = \mathbf{a}(v_c)\left[\mathbf{s}(v_p) + (1 - \mathbf{s}(v_p))\, e^{-\tau \mathbf{a}(v_c)}\right] . \quad (3)$$

This weight does the following: (1) it weights the edge by the area of the child node so that the smoothness term operates on all regions equally, (2) it weights the edge using

the region similarity $\mathbf{s}(v_p)$ and (3) it up-weights the edge for small regions to encourage them to be better connected to their parents, modulated by the parameter τ.

3.3. Cost function

We denote the image motions using a function $\mathbf{u} : \mathcal{V} \rightarrow \mathbb{Z}^2$, and we say that vertex v is matched to location $\mathbf{x}(v) + \mathbf{u}(v)$ in I_2. The cost function is defined over the graph structure for this displacement function $\mathbf{u}$:

$$\mathcal{C}(\mathbf{u}) = \lambda_0 \sum_{v \in \mathcal{V}} \mathcal{P}_v(\mathbf{u}(v)) + \lambda_1 \sum_{v \in \mathcal{V}} \mathcal{D}_v(\mathbf{u}(v)) + \sum_{(v_p, v_c) \in \mathcal{E}} \mathcal{S}_{v_p, v_c}(\mathbf{u}(v_p), \mathbf{u}(v_c)) \ . \quad (4)$$

Here, $\mathcal{P}_v(\mathbf{u}(v))$ is a prior term that encourages each node to have a small offset. The term $\mathcal{D}_v(\mathbf{u}(v))$ is a unary matching term that measures how well vertex v is matched, and $\mathcal{S}_{v_p, v_c}(\mathbf{u}(v_p), \mathbf{u}(v_c))$ is a smoothness term defined over the set of edges. The parameters λ_0 and λ_1 control the tradeoff of these terms.

We set $\mathcal{P}_v(\mathbf{u}(v)) = \mathcal{D}_v(\mathbf{u}(v)) = 0$ for all nodes that do not correspond to pixels at the base of the hierarchy. In other words, our unary terms affect only the leaves. The hierarchy thus is used only for enforcing smoothness. It is possible to add unary costs to internal nodes to incorporate region information, but we do not consider this here.

3.3.1 Spatial prior term

The spatial prior term encourages pixels to have small displacements. For this term, we fit a Cauchy distribution to the set of groundtruth offsets in the MPI-Sintel training dataset. The spatial prior term is then the negative log-likelihood of the associated Cauchy distribution with $\gamma_{prior} = 3.3942$:

$$\mathcal{P}_v(\mathbf{u}(v)) = \log \left[\pi \left(\mathbf{u}(v)^2 + \gamma_{prior}^2 \right) / \gamma_{prior} \right] \ . \quad (5)$$

3.3.2 Matching term

We use a data matching term that is composed of both a color and a gradient component. Let I_1 and I_2 be two images in the Lab color space. For color features, we construct a 5×5 array of cells, centered at a given pixel. The size of the cells can be varied in order to accumulate a larger or smaller contextual area. All pixel values within each cell are averaged, and this is applied to all three channels of the Lab color images, resulting in a 75-dimensional feature vector. Distances between feature vectors are computed using an L1 distance function. Let $[\cdot]_C$ be a function that maps an image to its 75-dimensional color feature vector. The cost function is then given by:

$$\mathcal{D}_v^{LAB}(\mathbf{u}(v)) = \| [I_1]_C(\mathbf{x}(v)) - [I_2]_C(\mathbf{x}(v) + \mathbf{u}(v)) \|_1 \ . \quad (6)$$

The gradient component of our data cost function uses SIFT features [21] computed densely at every pixel. Let $[\cdot]_S$ be a function that maps an image to its 128-dimensional SIFT features computed densely at each pixel location in Ω. We use an L1 cost function:

$$\mathcal{D}_v^{SIFT}(\mathbf{u}(v)) = \| [I_1]_S(\mathbf{x}(v)) - [I_2]_S(\mathbf{x}(v) + \mathbf{u}(v)) \|_1 \ . \quad (7)$$

The final cost function is a weighted linear combination of the color and SIFT costs:

$$\mathcal{D}_v(\mathbf{u}(v)) = \alpha \mathcal{D}_v^{LAB}(\mathbf{u}(v)) + (1 - \alpha) \mathcal{D}_v^{SIFT}(\mathbf{u}(v)) \ . \quad (8)$$

Note that the only desideratum of the distance function is that it be computationally efficient; it need not be a metric, differentiable, convex, or even continuous. This allows us to use SIFT features in dense optical flow without computing matches beforehand, and opens the door to more complex descriptors and distances.

3.3.3 Smoothness term

The smoothness term encourages vertices to have a similar offset to their parent. We use a weighted $L1$ penalty,

$$\mathcal{S}_{v_p, v_c}(\mathbf{u}(v_p), \mathbf{u}(v_c)) = \mathbf{w}(v_p, v_c) \| \mathbf{u}(v_p) - \mathbf{u}(v_c) \|_1 \ , \quad (9)$$

for v_p the parent of v_c, where $\mathbf{w}(v_p, v_c)$ is the associated edge weight as defined in Equation (3).

Note that this tree-based smoothness term is a significant departure from more traditional smoothing functions which are usually formulated on a graph that connects each image region to all of its neighbors. It has been previously observed that people's perceptual organization of objects in an image is hierarchical in nature [22]; we likewise argue that this tree-based smoothing term captures most of the salient features of the true motion field, while yielding a tractable optimization problem.

4. Optimization

Our model can be though of as a large "deformable parts" model (DPM) [9], which is a widely-used framework in object recognition. In a DPM, an object is represented by a tree that models how the parts of the object are connected to each other. Each node has a unary cost function that reflects its preference for various matches, and each edge in the tree is associated with a smoothness function that constrains the relative motion of the two parts. Although our model is more complex – our tree describes an entire image rather than a single object – the optimization problem is similar: in both cases the model is a tree-structured MRF and the label space captures the set of possible displacements.

The relationship between our model and a DPM allows us to leverage optimization techniques first developed in

the DPM literature. First, because our graph is a tree, the minimum-energy solution can be found in polynomial time using a generalization of the Viterbi algorithm [10]. Also, because we use an $L1$ distance function, the cost matrices at each node can be computed very efficiently using a linear-time distance transform [8].

To briefly summarize this procedure, the cost for each node is computed as a function of its children. This process begins at the leaf (pixel) nodes, and the cost matrices of each internal node are computed by applying distance transforms to the cost matrices of its children and summing them. We also compute and store a Voronoi array that specifies the optimal label for each vertex given a label for its parent. When the root of the tree is reached, the optimal cost for the entire model is found, and the labels of each vertex are set by backtracing back down the tree using the stored Voronoi arrays. For brevity, we omit further details of this procedure, which can be found in [9, 10].

4.1. Implementation details

Even though our tree-based MRF model can be optimally solved in polynomial time, a naive implementation would still be computationally demanding because of the number of nodes and the size of the label space. We thus make the following optimizations and approximations so that it can be computed on a standard computer in a reasonable amount of time with high accuracy. Note that although the result of these approximations is that the solutions are no longer exact, in practice we have found that the flow estimates produced are nearly indistinguishable from the true minima of the model.

Subsampling of superpixels We assume that the superpixels at the base of the segmentation hierarchy are sufficiently small that the optimal offsets of their constituent pixels will be very similar. This implies that their cost matrices will also be very similar, and so rather than computing a separate cost matrix for every pixel within a superpixel, we only compute the cost matrix for a small set of randomly-sampled constituent pixels. In practice, we use only 10 pixels per superpixel. Since each superpixel may contain hundreds or thousands of pixels, this significantly reduces the computational complexlity.

However, this creates a problem when backtracing down the tree to label each pixel. Because a cost matrix is not created for all pixels at the base of the hierarchy, they cannot all be directly labeled. We could circumvent this problem by computing the full cost matrix for each pixel only when it needs to be labeled at the end of the backtracing step, but this is still time consuming. Instead, we assume that a pixel's offset will be close to its parent's offset and only compute the cost in a small window around the parent's offset. In practice, we use a window that has a radius that is either 2 pixels or 20% of the magnitude of the parent's offset, whichever is larger. Because a large majority of pixels in many datasets have small motions, this is significantly more efficient than computing the full cost matrix for each pixel.

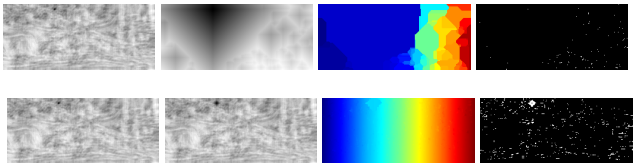
Subsampled cost matrices When performing the upward pass of optimization, we do not compute a full cost matrix for each node. Instead, we only compute every k^{th} pixel in a grid for some value of k. Because neighboring pixels will have similar costs, this results in a minimal decrease in accuracy while significantly speeding up the optimization. This strategy effectively reduces the number of labels on the upwards pass by only allowing each node to take every k^{th} label. However, at the final level of the downward pass, we compute a dense cost matrix for each pixel in the vicinity of its parent's optimal displacement and use this to determine its label. Thus, the size of the label space for the pixels is not reduced.

Sub-pixel localization Although our method is discrete, sub-pixel estimates can improve results. We do so in a similar way to [12]. At the pixel level of the downward pass of the optimization, a cost matrix for each pixel is computed as mentioned previously. Let (x, y) be the location of the minimum value in the pixel-level cost matrix. We fit a parabola to the cost values at locations $(x-1, y), (x, y), (x+1, y)$ and analytically locate the minimum. The y-direction is treated similarly by finding the minimum of a parabola fit to the values at $(x, y-1), (x, y), (x, y+1)$. Note that because (x, y) is a local minimum, the two quadratics will have well-defined minima near the location (x, y). The sub-pixel motion estimates are computed analytically with a negligible increase in runtime.

Compressing the cost matrices The values within the cost matrices need to be known only approximately. Rather than store the full matrix as double-precision floating point values, we store the minimum and maximum values in each matrix, scale the cost matrices to be between 0 and $2^8 - 1$, and store them as unsigned 8-bit integers.

Compressing the Voronoi matrices The Vornoi diagrams – which encode where every node should be placed as a function of its parent's offset – need to be computed and stored for each of the nodes in the graph and this can consume a significant amount of memory. In practice, we have observed that these diagrams often conform to one of two patterns (Figure 2). If the smoothness cost is small, the Voronoi diagram will be dominated by only a few entries corresponding to a few dominant local minima of the cost matrix and all other cost matrix values can be discarded without altering the resulting Voronoi diagram. In this case, we store only these entries of the cost matrix and the Voronoi diagram can then be recomputed when it is needed.

Alternatively, if the smoothness constraint is relatively



(a) Cost matrix (b) Distance transform (c) Voronoi diagram (d) Indices stored

Figure 2: The Voronoi diagram encodes the optimal label of a node given its parent's location. It can be compressed two ways. The last column shows the entries that need to be stored of the cost matrix (first row) or Voronoi diagram (second row). **Top row:** When the smoothness cost is small, the Voronoi diagram is dominated by a few entries. Only a few locations of the cost matrix are needed to reproduce the Voronoi diagram. **Bottom row:** When the smoothness cost is large, most indices of the Voronoi diagram are just the index of the location itself, and we only store the indices which differ.

high then the best motion of a node will usually be similair to that of its parent. In this case we store a sparse matrix encoding the entries where the child's displacement differs from that of its parent. The full, non-sparse Voronoi diagram can then be constructed when it is needed.

At runtime the system determines which compression scheme is most effective for each node and applies it.

5. Multi-frame optical flow

For two-frame optical flow, large displacements can make it difficult or even impossible to find correct correspondences, such as when an object moves out of frame. In [16] a simple method was proposed to incorporate information from multiple frames. This is done by assuming that objects move linearly, from inertia, over a short time span. Multiple "inertial" estimates of the flow are formed from nearby frames and subsequently fused using a classifier.

In our global optimization framework, inertial estimates can be directly incorporated into the data cost term without changing the optimization procedure at all (Figure 3). Let $\mathcal{D}_v^{t+1}(\mathbf{u}(v))$ be the data term for pixel node v with respect to frame $t+1$, which is just the standard two-frame data cost function. Similarly, let $\mathcal{D}_v^{t-1}(-\mathbf{u}(v))$ be the cost as projected onto frame $t-1$ and let $\mathcal{D}_v^{t+2}(2\mathbf{u}(v))$ be the cost with respect to frame $t+2$. If a vertex is occluded, it may have a high cost with respect to frame $t+1$, but might match better to one of the other frames. In other words, we want vertex v to match to *one* of the three frames, but not necessarily all. We then define

$$\mathcal{D}_v(\mathbf{u}(v)) = \min \begin{cases} \mathcal{D}_v^{t+1}(\mathbf{u}(v)), \\ \mathcal{D}_v^{t-1}(-\mathbf{u}(v)) + \beta, \\ \mathcal{D}_v^{t+2}(2\mathbf{u}(v)) + \beta\ , \end{cases} \tag{10}$$

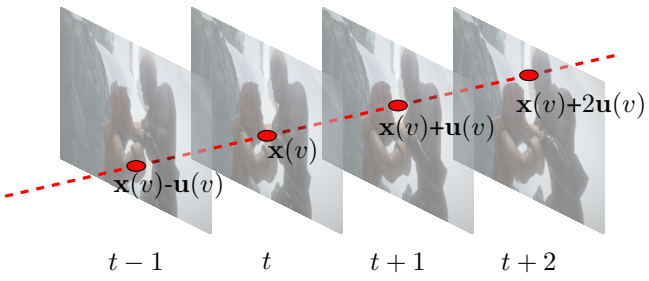


Figure 3: Depiction of our multi-frame term. Rather than only computing the data cost from frame t to $t+1$, the offset is extended to other adjacent frames, and the data terms are combined into a single cost matrix. The intuition is that if a node is occluded, it may match better to another adjacent frame, assuming linear motion.

where $\beta > 0$ biases the cost towards matching to frame $t+1$. This data cost is then used directly within the optimization. Note that it would be difficult to use this method within a continuous optimization framework because the cost function is no longer differentiable everywhere and has many more local minima.

6. Experiments

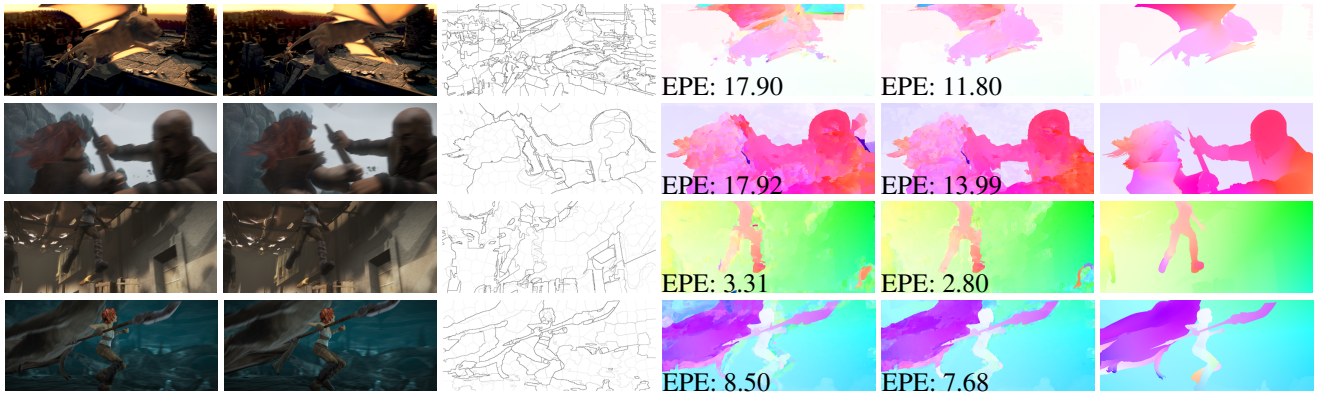
Our algorithm is denoted by HCOF, while the HCOF+multi variant uses multiple frames as described in Section 5.

Parameters Parameters for all experiments in this paper were set using the Final training dataset of MPI-Sintel [6]. Parameters were optimized using a small grid search. We set $\alpha = 0.15$, $\lambda_0 = 1$, $\lambda_1 = 2$, $\eta_0 = 50$, $\eta_1 = -0.05$, $\tau = 0.01$, and $\beta = 40$ when multiple frames were used. The maximum offset was set to 200 pixels, which accounts for 99.9% of all pixels on the MPI-Sintel training dataset. Note that this results in a label space with 160,000 labels. On the upward pass, the data cost matrices were sampled every 3 pixels. The SIFT features had a cell size of 5 pixels and the Lab color features had a cell size of 3 pixels. For SLIC superpixels, the region size parameter was set to 50 and the regularization parameter was also set to 50.

6.1. MPI-Sintel

The MPI-Sintel dataset [6] is a difficult optical flow dataset created using an open-source 3D computer-generated film. The dataset consists of over 1000 images and contains difficulties such as large displacements, significant occlusions, lighting variation, and motion blur.

Results for several images chosen from the Final training set of MPI-Sintel are show in Figure 4. The global nature of the optimization coupled with the underlying superpixel segmentation results in flow estimates with crisp motion



(a) First image (b) Second image (c) Segmentation (d) HCOF (e) HCOF+multi (f) Groundtruth

Figure 4: Results on several images from the Final dataset of MPI-Sintel. The flow images are colored with respect to the maximum groundtruth displacement.

Table 1: Evaluation on the Final training dataset of MPI-Sintel. The use of inertial estimates improves results.

	EPE	s0-10	s10-40	s40+
HCOF	5.882	3.163	6.786	17.596
HCOF+multi	5.265	2.800	5.928	15.892

boundaries as opposed to the oversmoothed motions produced by many schemes based on continuous optimization.

We have found that the use of multiple inertial estimates of optical flow in the data cost can significantly improve results, especially in occluded regions. This is illustrated quantitatively in Table 1 which summarizes the results obtained with both HCOF and HCOF+multi on the Final training dataset.

Quantitative results on the test set for the method HCOF+multi are shown in Table 2. Our method is competitive with many modern methods, although the errors are still below the state-of-the-art. A significant fraction of the error can be attributed to segmentation errors. When we evaluated the method on the training set using a segmentation derived from the ground truth motion, the endpoint error was reduced by 15%. This suggests that future work could consider strategies that refine the segmentation tree as the optimization proceeds.

Our results were computed on a computer with a 2 GHz Intel Xeon processor and 4 GB of memory. The computation took about 10 minutes when multiframe estimates were not used and about 25 when they were, for the full-size 1024×436 images. The computation time could be substantially reduced by using simpler features and smaller images. We also note that several parts of our algorithm, such as computing the data cost for each pixel, can readily be parallelized.

Table 2: Evaluation on the Final test dataset of MPI-Sintel. We report endpoint error for all pixels, and also based on the groundtruth speed.

	EPE	s0-10	s10-40	s40+
EpicFlow [26]	6.285	1.135	3.727	38.021
TF+OFM [16]	6.727	1.512	3.765	39.761
DeepFlow [31]	7.212	1.284	4.107	44.118
AggregFlow[12]	7.329	1.241	4.296	44.858
ChannelFlow[28]	8.835	1.292	5.349	54.648
LDOF [5]	9.116	1.485	4.839	57.296
AnisoHuber.L1 [32]	11.927	1.155	7.966	74.796
HCOF+multi	8.799	1.682	5.786	51.363

6.2. Synthetic Dataset

In order to illustrate the issues that often occur in the face of large motions, we constructed a simple synthetic dataset using the imagery from the Final dataset of MPI-Sintel. An example of an image pair from this dataset is shown in Figure 5. For each such pair a background is generated by choosing an image uniformly at random and selecting a random 256×256 square patch from that image. We then generate an "object" by selecting a random 32×32 patch from another image. The object patch is placed at a random location on the background and is then translated in a random direction by a specified offset distance. For a given offset distance, we evaluate each algorithm averaged over 100 such random images.

We deliberately chose a relatively simple motion stimulus to illuminate how various schemes handle layered motions with large displacements. Adding more layers, more motions and more objects would not fundamentally alter the results.

(a) First image (b) Second image (c) Groundtruth

Figure 5: An example of an image from our synthetic dataset. A random 256×256 patch is used as a background while another 32×32 "object" patch is moved a given offset between images. In this example, the offset is 50 pixels.

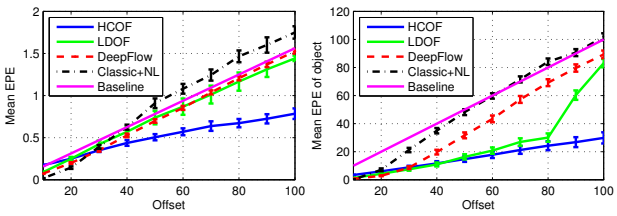
We compare our own algorithm, HCOF, to several other modern approaches for which code was readily available: Classic+NL [30], LDOF [5], and DeepFlow [31]. Classic+NL is a modern implementation of a coarse-to-fine variational approach. Large Displacement Optical Flow (LDOF) incorporates sparse feature matching into the optimization in order to better deal with large displacements. Finally, DeepFlow is a top-performing algorithm on MPI-Sintel that uses a more advanced feature matching term to account for deformation. For all approaches, we used their default parameters, except for DeepFlow where we used the "improved settings" as documented in their code. We also compare to the baseline method of predicting a zero-valued flow.

Results on this dataset are given in Figure 6. We show the mean endpoint error for each method as averaged over 100 random images, and the offset is varied from a minimum of 10 pixels to a maximum of 100 pixels. The error is evaluated for both the full image as well as only the object patch itself.

In all cases, the error increases roughly linearly as the offset is increased. For offsets of less than 20 pixels, HCOF is outperformed by the other coarse-to-fine continuous methods which are able to achieve better sub-pixel accuracy and are not affected by segmentation error. However, HCOF is significantly more robust to large displacements. For offsets of 100 pixels, our method achieves errors several times lower than other approaches. The reason for this is that the magnitude of the offset has little effect on the accuracy of our method because it uses a discrete, global optimization. Additionally, segmentation error is minimal on this dataset.

6.3. Tracking Dataset

To demonstrate that the issues illustrated in Section 6.2 do in fact occur in real images we evaluated our algorithm on the Hotel sequence from the BIWI Walking Pedestrians dataset [24], which is a video of a city sidewalk taken from an overhead camera. The video was taken at 25 frames per



(a) EPE for entire image (b) EPE for object region

Figure 6: Endpoint error on a synthetic dataset where an object is translated a varying amount. The Baseline method is a zero-flow estimate. The endpoint error is averaged over 100 images and we show the standard error as error bars. In (a), endpoint error is averaged over the entire image, while in (b) we plot the error for only the object itself.

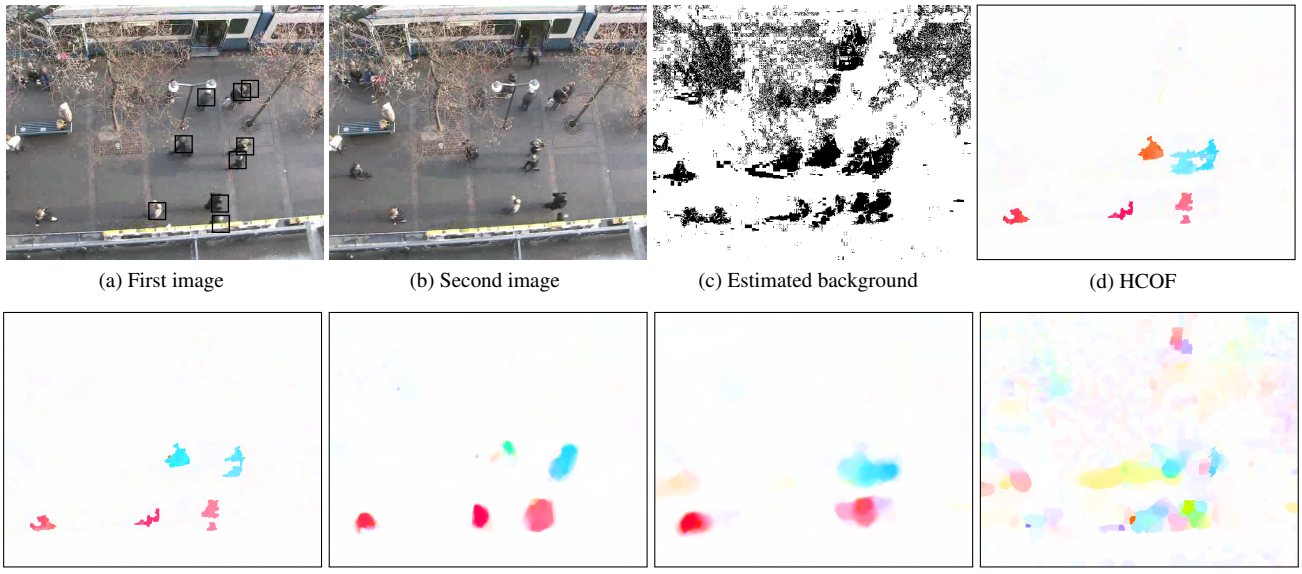
second and annotations are provided for every 10th frame indicating the positions of people visible in the scene. We further subsample every other annotated frame so that the dataset contains both large and small motions, resulting in a total of 572 annotated frames each with a resolution of 720×576 pixels. Two frames from this dataset are shown in Figure 7.

We evaluate flow methods on this dataset in two ways. First, we have sparse annotations of the locations of people. For these points we calculate the groundtruth offsets and compare them to the results from each method. However, this may penalize methods that successfully track most of a person but miss the one annotated pixel, and so instead we find the best estimated flow value within a 41×41 box around each annotated point, as shown in Figure 7. This evaluation is done independently on all 2623 pedestrian annotations in the dataset.

We also evaluate methods by noting that the camera is stationary and that much of the image will have zero optical flow. We determine the stationary pixels in each frame by thresholding the magnitude of the intensity difference between frames at 0.05. We evaluate the flow on these background pixels separately.

Qualitative results on a pair of frames from this dataset are shown in Figure 7, and quantitative results are given in Table 3. For the annotated tracks, the results are presented using average EPE and are also divided into bins based on the magnitude of the groundtruth motion, similar to the results for MPI-Sintel (Section 6.1). The last column of the table also shows the mean EPE over all the background pixels.

LDOF performs the best of all continuous methods, but for large offsets HCOF+multi has a significantly lower error than other methods. DeepFlow may underperform on this dataset as compared to MPI-Sintel because the objects that are moving are relatively small and DeepFlow does



(a) First image (b) Second image (c) Estimated background (d) HCOF

(e) HCOF+multi (f) LDOF (g) DeepFlow (h) Classic+NL

Figure 7: An example of a result from the Tracking dataset. In (a), the black boxes indicate the people for which annotations are provided and the extent of the boxes is the area over which we look for the lowest endpoint error for each person. In (c), we show the estimated background pixels, for which we assume the flow is zero in our evaluation. The results of different methods are given in (d)-(h). The results of our method HCOF are much more localized.

not compute a fully-dense feature matching due to memory constraints. Overall, HCOF+multi outperforms all other methods on this dataset. We also find that HCOF and HCOF+multi are significantly more accurate on the background pixels that have zero flow than all methods except for Classic+NL, which has a low error only because it fails to correctly estimate the motions of the people. All other methods tend to pull part of the background along with the objects and are less able to localize the objects themselves. This can be seen visually in the results in Figure 7. Note, for example, that all methods other than HCOF are unable to separate the two people walking next to each other at the bottom of the image.

Table 3: Evaluation on the tracking dataset. The left section of the table shows the endpoint error over all annotated tracks, both overall and divided based on the magnitude of the groundtruth motion vectors. The last column measures the endpoint error over estimated background pixels.

	Annotated Tracks				Background
	EPE	s0-10	s10-40	s40+	EPE
LDOF [5]	3.031	0.654	**1.954**	4.333	0.738
DeepFlow [31]	13.048	0.862	4.751	20.265	0.730
Classic+NL [30]	35.220	0.821	16.705	54.631	**0.385**
HCOF	2.896	**0.603**	2.321	4.060	0.580
HCOF+multi	**2.561**	0.678	2.948	**3.349**	0.430

7. Summary

In this paper we have described a novel approach for estimating optical flow that leverages a hierarchical segmentation of the image. To the best of our knowledge this is the first time that a discrete, global optimization approach has been successfully applied to large scale optical flow problems involving hundreds of thousands of motion labels.

The computational innovations that we have described make it possible to solve for per-pixel flows using a reasonable amount of computation and memory. The experimental results suggest that the proposed tree-based smoothing cost captures most of the salient features of the actual flow field and that a global optimization approach can improve performance in situations that involve large motions.

Future work will look at refining the scoring function associated with the superpixels to allow for more complex deformation models. We also plan to explore ideas for refining the segmentation tree over the course of the procedure to improve the final motion estimates.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012. 2

[2] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *CVPR*, pages 182–182. IEEE, 2006. 2

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 2

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *ECCV*, 2004. 1

[5] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, 33(3):500–513, 2011. 6, 7, 8

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625. Springer, 2012. 1, 5

[7] M. Donoser and D. Schmalstieg. Discrete-continuous gradient orientation estimation for faster image segmentation. *CVPR*, 2014. 2

[8] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell University, 2004. 4

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010. 1, 3, 4

[10] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005. 4

[11] X. Feng, C. K. Williams, and S. N. Felderhof. Combining belief networks and neural networks for scene segmentation. *PAMI*, 24(4):467–483, 2002. 2

[12] D. Fortun, P. Bouthemy, and C. Kervrann. Aggregation of local parametric candidates with exemplar-based occlusion handling for optical flow. *arXiv preprint arXiv:1407.5759*, 2014. 4, 6

[13] T. Goldstein, X. Bresson, S. Osher, and A. Chambolle. GLOBAL MINIMIZATION OF MARKOV RANDOM FIELDS WITH APPLICATIONS TO OPTICAL FLOW. *Inverse Problems & Imaging*, 6(4), 2012. 2

[14] H. Ishikawa. Exact optimization for markov random fields with convex priors. *PAMI*, 25(10), 2003. 2

[15] J. H. Kappes, M. Speth, G. Reinelt, and C. Schnorr. Towards efficient and exact MAP-inference for large scale discrete computer vision problems via combinatorial optimization. In *CVPR*. IEEE, 2013. 2

[16] R. Kennedy and C. J. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. *EMMCVPR*, 2015. 1, 5, 6

[17] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2307–2314. IEEE, 2013. 2

[18] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1568–1583, 2006. 1

[19] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 508–515. IEEE, 2001. 2

[20] C. Lei and Y.-H. Yang. Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1562–1569. IEEE, 2009. 2

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 3

[22] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE, 2001. 3

[23] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988. 1

[24] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 261–268. IEEE, 2009. 7

[25] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers. A convex formulation of continuous multi-label problems. In *ECCV*, pages 792–805. Springer, 2008. 2

[26] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 6

[27] J. Reynolds and K. Murphy. Figure-ground segmentation using a hierarchical conditional random field. In *Canadian Conference on Computer and Robot Vision*, pages 175–182. IEEE, 2007. 2

[28] L. Sevilla-Lara, D. Sun, E. G. Learned-Miller, and M. J. Black. Optical flow estimation with channel constancy. In *Computer VisionECCV 2014*, pages 423–438. Springer, 2014. 6

[29] E. Strekalovskiy, B. Goldluecke, and D. Cremers. Tight convex relaxations for vector-valued labeling problems. In *ICCV*, pages 2328–2335. IEEE, 2011. 2

[30] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. *CVPR*, 2010. 1, 7, 8

[31] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. *ICCV*, 2013. 6, 7, 8

[32] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic huber-l1 optical flow. In *BMVC*, page 3, 2009. 6

[33] S. Wu, X. He, H. Lu, and A. L. Yuille. A unified model of short-range and long-range motion perception. In *Advances in Neural Information Processing Systems*, pages 2478–2486, 2010. 2

[34] L. Zhu, Y. Chen, Y. Lin, C. Lin, and A. Yuille. Recursive segmentation and recognition templates for image parsing. *PAMI*, 34(2):359–371, 2012. 2