Optimizing Motion Estimation with Linear Programming and Detail-Preserving Variational Method

Hao Jiang, Ze-Nian Li, and Mark S. Drew
School of Computing Science, Simon Fraser University,
Vancouver, British Columbia, Canada V5A 1S6
{hjiangb,li,mark}@cs.sfu.ca

Abstract

In this paper, we propose a novel linear programming based method to estimate arbitrary motion from two images. The proposed method always finds the global optimal solution of the linearized motion estimation energy function and thus is much more robust than traditional motion estimation schemes. As well, the method estimates the occlusion map and motion field at the same time. To further reduce the complexity of even a complexity-reduced pure linear programming method we present a two-phase scheme for estimating the dense motion field. In the first step, we estimate a relatively sparse motion field for the edge pixels using a non-regular sampling scheme*, based on the proposed linear programming method. In the second step, we set out a detail-preserving variational method to upgrade the result into a dense motion field. The proposed scheme is much faster than a purely linear programming based dense motion estimation scheme. And, since we use a global optimization method — linear programming — in the first estimation step, the proposed two-phase scheme is also significantly more robust than a pure variational scheme.*

1 Introduction

Motion estimation is a key technique for vision applications such as object segmentation, object recognition, tracking, and 3D scene reconstruction. Optical flow based methods have been intensively studied for the small scale motion estimation problem, e.g. the motion analysis of successive frames in a high frame-rate video. Horn and Schunck [1] formulated the optical flow problem as an energy minimization problem and presented a steepest descent iterative scheme. In their method, a quadratic norm is used in the smoothing term. Another well-known method to calculate optical flow is that of Lucas and Kanade [2], in which a weighted least square norm is used to fit the local first order optical flow constraint. Different optical flow methods are evaluated in [3]. A recent optical flow method based on the structure tensor and a local parameter model is studied in [4].

Besides the optical flow formulation, another method to study the motion estimation problem is based on the explicit matching concept. In this approach, motion estimation corresponds to finding a mapping from the first image to the second image. The mapping can be restricted to some specific model, such as affine motion estimation. Another kind of mapping is model-less mapping, in which there is no explicit motion constraint. Model based matching involves fewer parameters and is more robust for specific applications, while model-less matching has more flexibility and can be easily adapted to different applications. Model-less matching has attracted a great deal of interest in recent years. Several model-less matching schemes for horizontal motion estimation for rectified images in the stereo problem have been intensively studied [5, 6, 8, 9, 10]. Also, a max-posterior probability based matching method [11] was presented for the motion analysis problem.

Motion estimation or matching problems can be generalized as searching problems in a given parameter domain. One of the key problems is how to search the feasible parameter space constrained by a variety of constraints such as smoothing and occlusion conditions, or other model constraints, to obtain an optimal solution which generates the best feature consistency between two images. There are two classes of search methods used in motion analysis algorithms. The first class of methods start from some initial guess and try to find a local minimum. Most optical flow schemes belongs to this class. The merits of the local search schemes are that they have relatively low complexity compared to some other searching schemes. The problem, however, is that the quality of the solution is greatly affected by the initial value selected — and in fact it is usually difficult to get a good initial guess in most real applications. The second class of methods search the feasible space and get a globally best solution. E.g., the graph-cut scheme [5, 6, 7, 8], a global optimization method, has been successfully applied to the stereo problem. The graph cut scheme has also been applied to motion estimation with a strong approximation scheme [12]. Other global optimization schemes used for matching problems include stochastic methods such as simulated annealing [13]. Global optimization is usually much more robust than local searching schemes.

Besides the above global searching schemes, a linear programming approach presents another general framework for global optimization problems. Linear programming has been applied to match feature points in different views for a fixed number of correspondences of feature points [14], for estimating affine motion and homography in two different views [15], and for pose estimation under affine projection

[16]. A linear programming method is also presented for the labeling problem [17], and this could be used for the motion estimation problem. The problem with the latter scheme is that the large number of variables makes it difficult to apply in real applications.

In this paper, we propose a new linear programming based method to estimate relatively *large scale motion*. Here, motion can have both x and y freedom in the two images. The proposed method involves far fewer variables and does not need a rounding process for motion estimation. Our formulation is a model-less approach. The proposed method can also *estimate the occlusion map and motion field at the same time*. By using a linear programming formulation, the proposed method can be used to globally optimize the objective function efficiently with a method such as the simplex method. To alleviate the high-complexity problem of a pure linear programming method (although it is already much less than for a nonlinear global optimization) we suggest a two-phase scheme for estimating a dense motion field. In the first step, we estimate a relatively sparse motion field (cf. [1, 2]) using the edge pixels and small set of supporting non-edge pixels, here, based on a proposed new linear programming method. In the second step, we propose a detail-preserving partial differential equation (PDE) method to upgrade the result into a dense motion estimation. Comparing to Horn and Schunck's variational formulation, the proposed method can adapt to the local structures of the motion field. The proposed two-phase scheme is much faster than a purely linear programming based dense motion estimation scheme. As well, since we use a globally optimizing method (linear programming) in the first estimation step, the proposed two-phase scheme is more robust than a pure variational or PDE scheme.

The arrangement of the paper is as follows. In §2 we first propose a *nonlinear* formulation of the motion estimation problem. Then, we study how to convert the nonlinear optimization problem into an equivalent linear programming formulation. In §3 we present a two-phase method for dense motion field estimation based on a sparse linear programming formulation and a detail-preserving variational method. In §4 we present the results of the proposed method applied to the dense motion estimation and stereo problems. We conclude the paper in §5.

2 Motion Estimation Based on Linear Programming

We study the problem of motion estimation based on the image matching concept. The image matching problem can be stated as: Given a pixel set S in the first image, we need to find a function $\mathcal{M}$ such that $s' = \mathcal{M}(s)$, where $s \in S$ and s' is the corresponding matched pixel in the second image. We require that $\mathcal{M}$ is a function such that for each pixel in the first image I_1, there is one and only one matching pixel in the second image I_2 . In this formulation, image I_1 and I_2 are not symmetric, since for each pixel in I_2, there could be more than one matching pixel in I_1. The non-symmetric definition is reasonable for real applications since the limited sampling rate and some specific motion may lead to several pixels in the first image mapping to the same pixel in the second image. To differentiate the roles of the two images in the matching process, we define the first image as the *reference* image I_r and the second the *matching* image I_m. Besides motion estimation, we usually also prefer to be able to estimate whether a pixel in the reference image is occluded in the matching image. Estimating the *occlusion map* is usually a much more difficult problem and has been ignored in many motion estimation algorithms, especially for cases where the occlusion is small, for example for high sampling-rate videos. For wide baseline matching problems, a good estimation of the occlusion map can usually greatly improve the shape reconstruction result at the occluding boundaries. Thus in summary, for the motion estimation problem we need to estimate the motion for non-occluded pixels and the corresponding occlusion map.

In this section, we present a method based on a linear programming formulation to estimate the motion map and occlusion map at the same time. We first formulate the motion and occlusion estimation problem as a nonlinear optimization problem. Then we *convert the nonlinear optimization problem into an equivalent linear programming formulation* based on linear approximations and variable relaxation.

2.1 Nonlinear Motion and Occlusion Optimization

The estimation of motion and occlusion can be formulated as the following minimization problem:

$$\min_{d^{(x)},d^{(y)},g} \sum_{(x,y)\in S} \left[C_{x,y,d^{(x)}_{x,y},d^{(y)}_{x,y}} \cdot \overline{g}_{x,y} + C_o \cdot g_{x,y} \right] + \sum_{[(x_1,y_1),(x_2,y_2)]\in\mathcal{N}} \{\lambda_{x_1,y_1,x_2,y_2}[|d^{(x)}_{x_1,y_1} \cdot \overline{g}_{x_1,y_1} - d^{(x)}_{x_2,y_2} \cdot \overline{g}_{x_2,y_2}| + |d^{(y)}_{x_1,y_1} \cdot \overline{g}_{x_1,y_1} - d^{(y)}_{x_2,y_2} \cdot \overline{g}_{x_2,y_2}|] + \mu_{x_1,y_1,x_2,y_2}|g_{x_1,y_1} - g_{x_2,y_2}|\}$$

In the optimization problem, we search for optimal motions in the horizontal and vertical directions, represented as $d^{(x)}_{x,y}$ and $d^{(y)}_{x,y}$ respectively, and the occlusion map $g_{x,y}$ for each site in the set S in the reference image such that the energy defined is minimized. The occlusion map $g_{x,y}$ is a binary function defined such that if pixel (x, y) is occluded in the matching image, $g_{x,y} = 1$, and otherwise $g_{x,y} = 0$. The complement of $g_{x,y}$ is $\overline{g}_{x,y}$, equaling $1 - g_{x,y}$. The energy function consists of two parts. The first part defines the cost of the motion estimation or occlusion decision at a site, in which $C_{x,y,d^{(x)},d^{(y)}}$ is the cost function for assigning the motion $(d^{(x)}, d^{(y)})$ to site (x, y) and C_o is the cost of labeling a site as occluded. The second part of the energy function is a regularity term that penalizes the discontinuity of neighboring motions or occlusion decisions. $\mathcal{N}$ is the set of neighboring sites in S and the coefficients λ and μ control the smoothness of the motion field and the occlusion map estimated. In the regularity term, we choose

the L_1 norm, which has better properties for preserving discontinuity than does the L_2 norm. Note that we include subscripts for both μ and λ to indicate that in our scheme we can make μ_{x_1,y_1,x_2,y_2} and $\lambda_{x_1,y_1,x_2,y_2}$ adaptive to the coupling strength of the two sites at (x_1, y_1) and (x_2, y_2), which is necessary if we use non-regular grids. We discuss how to select the coefficients dynamically in more detail in § 3.1. In motion analysis for digital images, the set S is a discrete set which corresponds to the position of the image pixels. But the motion $d^{(x)}$ and $d^{(y)}$ can be fractional numbers.

In the above energy minimization formulation, if a pixel and its neighbors are not occluded from the first image to the second image, the functional minimizes the motion estimation cost and penalizes motion discontinuities for neighboring sites. If the pixel is occluded in the second image, the functional minimizes the occlusion labeling cost and penalizes discontinuities for neighboring occlusion labels. This formulation also penalizes large motions at the occlusion and non-occlusion boundaries, which does not pose a problem because of the discontinuity preserving properties of the L_1 norm. The merit of this optimization configuration is that it facilitates the linear programming formulation. We wish occluded pixels to take on $g_{x,y} = 1$. The selection of cost C_0 should therefore follow the following rules. In the ideal case, for an occluded pixel at (x, y) we will need to make the cost $C_{x,y,d^{(x)},d^{(y)}} > C_0$ for all possible $(d^{(x)}, d^{(y)})$, so that the pixels will be labeled as occluded. On the other hand, for non-occluded pixels we need to make $C_{x,y,d_*^{(x)},d_*^{(y)}} < C_0$ for the actual motion $(d_*^{(x)}, d_*^{(y)})$, such that the pixel will be labeled as non-occluded. Based on this arrangement, if a pixel is occluded the minimization tends to assign 1 to $g_{x,y}$; otherwise the scheme will find a best motion estimation and assign 0 to $g_{x,y}$. In actuality, the above conditions cannot be completely satisfied but usually can be well approximated. Since we include smoothing regularity terms, our scheme in fact works quite well in actual situations.

2.2 Conversion to Linear Programming

The above energy optimization problem is nonlinear, so it is difficult to find a global optimization solution in this original form. Local searching schemes are usually not robust for the problem in which the displacements of pixels are relatively large without a good initial value estimate. We study how to cast the problem as linear programming, which is still globally optimized, by linear approximation and variable relaxation.

We first relax $g_{x,y}$ into a continuous function in the range $[0, 1]$. In this relaxation form, $g_{x,y}$ becomes a "soft" decision or likelihood of whether a pixel is occluded, instead of the binary hard decision in the original formation. To convert a soft decision into a hard decision a thresholding scheme can be applied. For example, a threshold of 0.5 is usually used in the threshold process. The merit of relaxation of $g_{x,y}$ into a continuous function is that we can convert the problem into a linear programming problem, not a harder mixed-integer programming problem.

To linearize the first term, we use the following scheme. We select a basis $\mathcal{B}_{i,j}$ for the displacement of each site (i, j), e.g. the positions of the vertices of the convex hull of the cost function over a 40×40 window, or even larger. Then the displacement $(d_{i,j}^{(x)}, d_{i,j}^{(y)})$ can be represented as a linear combination of the displacement basis as $(d_{i,j}^{(x)}, d_{i,j}^{(y)}) = \sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} \cdot (m, n)$. The labeling cost of motion $(d_{i,j}^{(x)}, d_{i,j}^{(y)})$ can be then approximated by the linear combination of the motion cost of the base motion costs $C_{i,j,\mathsf{x}\{\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n}\cdot(m,n)\},\mathsf{y}\{\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n}\cdot(m,n)\}} \simeq \sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} \cdot C_{i,j,m,n}$, where function $\mathsf{x}(\{x, y\}) = x$, $\mathsf{y}(\{x, y\}) = y$. We also further set constraints $\xi_{i,j,m,n} \geq 0$ and $\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} = 1$ for each site (i, j), so as to constrain the space spanned by the basis to the convex hull of the basis vectors. (Clearly, if $\xi_{i,j,m,n}$ are constrained to be 1 or 0, the above representation is exact.) Note that $d_{i,j}$ is *not* constrained to the basis motions, but can be any convex combination. To linearize the regularity terms in the nonlinear formulation we can represent a free variable by the difference of two nonnegative auxiliary variables and introduce the summation of the auxiliary variables into the objective function. If the problem is properly formulated, when the linear programming problem is optimized the summation will approach the absolute value of the free variable [18]. Below, we discuss in more detail how to structure the problem to achieve these goals.

Based on this linearization process, a linear programming formulation of the problem can be stated as

$$\min : \mathcal{I} = \sum_{(i,j)\in S} \sum_{(m,n)\in\mathcal{B}_{i,j}} C_{i,j,m,n} \cdot \xi_{i,j,m,n} + \sum_{(i,j)\in S} C_o \pi_{i,j} + \sum_{[(i,j),(k,l)]\in\mathcal{N}} \lambda_{i,j,k,l} (dx_{i,j,k,l}^{+} + dx_{i,j,k,l}^{-} + dy_{i,j,k,l}^{+} + dy_{i,j,k,l}^{-}) + \sum_{[(i,j),(k,l)]\in\mathcal{N}} \mu_{i,j,k,l} (\pi_{i,j,k,l}^{+} + \pi_{i,j,k,l}^{-})$$

subject to:

$$\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} + \pi_{i,j} = 1, \forall (i, j) \in S$$

$$\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} \cdot m = dx_{i,j}, \forall (i, j) \in S$$

$$\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} \cdot n = dy_{i,j}, \forall (i, j) \in S$$

$$dx_{i,j} - dx_{k,l} = dx_{i,j,k,l}^{+} - dx_{i,j,k,l}^{-}, \forall\, [(i, j), (k, l)] \in \mathcal{N}$$

$$dy_{i,j} - dy_{k,l} = dy_{i,j,k,l}^{+} - dy_{i,j,k,l}^{-}, \forall\, [(i, j), (k, l)] \in \mathcal{N}$$

$$\pi_{i,j} - \pi_{k,l} = \pi_{i,j,k,l}^{+} - \pi_{i,j,k,l}^{-}, \forall\, [(i, j), (k, l)] \in \mathcal{N}$$

with bounds:

$$\xi_{i,j,m,n} \geq 0$$
$$\pi_{i,j} \geq 0$$
$$\Delta_{x\max} \geq dx_{i,j} \geq \Delta_{x\min}$$
$$\Delta_{y\max} \geq dy_{i,j} \geq \Delta_{y\min}$$
$$dx^+_{i,j,k,l}, dx^-_{i,j,k,l} \geq 0$$
$$dy^+_{i,j,k,l}, dy^-_{i,j,k,l} \geq 0$$
$$\pi^+_{i,j,k,l}, \pi^-_{i,j,k,l} \geq 0$$

where $\xi_{i,j,m,n}$ and $\pi_{i,j}$ are the real-valued motion weighting and occlusion variables for site (i,j), respectively. Since we wish to carry out linear programming using only nonnegative variables, we compose reals from nonnegative pairs, e.g. $dx = (dx^+ - dx^-)$.

In $\mathcal{I}$, $C_{i,j,m,n}$ is selected based on the linearization scheme; $\mathcal{B}_{i,j}$ is the basis set for site (i,j); and $[\Delta_{x\min}, \Delta_{x\max}]$ and $[\Delta_{y\min}, \Delta_{y\max}]$ are the searching ranges of the motion in the x and y directions respectively. $dx_{i,j}$ and $dy_{i,j}$ are the motions at site (i,j) in the x and y directions. $\xi_{i,j,m,n}$ are the coefficients of the motion bases — clearly, the motion in the x direction $dx_{i,j}$ can be represented as the linear combination $\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} \cdot m = dx_{i,j}$. Similarly the motion in the y direction can be represented as $\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} \cdot n = dy_{i,j}$. The condition $\sum_{(m,n)\in\mathcal{B}_{i,j}} \xi_{i,j,m,n} + \pi_{i,j} = 1, \forall (i,j) \in S$ accounts for the coefficients $\overline{g}$ in the first and third term in the nonlinear optimization formulation. It should be noted that motions dx and dy in the linear programming formulation correspond to the $d^{(x)}\overline{g}$ and $d^{(y)}\overline{g}$ respectively, which are the real motions in the non-occlusion area and vanish in the occlusion area. This projected motion is well defined everywhere in the image. In the following sections, motion will be discussed in the sense of projected motion.

It is not difficult to show that the third and fourth term in the minimization equal the absolute value of the neighborhood motion field difference and occlusion map difference. Therefore the linear programming formulation is equivalent to the general nonlinear programming formulation if the linearization assumption is fulfilled. In real applications, the linear programming formulation is an excellent approximation of the original nonlinear optimization problem.

The above linear programming formulation degenerates to the local searching method if the smoothing terms are eliminated, and degenerates into the formulation with occlusion ignored if the occlusion cost term and penalty terms are removed. Note that it is not difficult to extend the formulation into a solution to the *general* labeling problem: this could have great use in the object tracking and segmentation and 3D reconstruction problems.

3 A Two-phase Method for Dense Motion Field Estimation

The above linear programming method guarantees global optimization of the problem but is usually too complex to be able to apply directly for estimation of a *dense* motion field with the computational power of current hardware. Since motion fields usually have much less high frequency content than the gray levels of images, the sampling rate of the motion field can be much lower than the sampling rate of the original images, and this implies that we can estimate a sparser motion field without forfeiting estimation quality. In this section, we present a sparse motion field estimation scheme. We only estimate the motion for the important points of the images (cf. [1, 2]). In this paper, we identify edge pixels and small set of supporting points as the important points. Edge pixels carry less ambiguity for the motion analysis problem than do pixels in textureless parts of the image and therefore should be able to actually improve the motion estimation result. First, we discuss a *non-regular sampling scheme* to estimate a sparse motion field. Then, we present a PDE-based method to upgrade the result into a dense motion field with resolution equal to that of the original images.

3.1 Sparse Estimation Based on Linear Programming Method

We first extract the edges by using the Prewitt edge detector. Then we randomly select ρ of the total number of edge pixels, where ρ is in $(0,1]$. Typical ρ is from 0.1 to 0.5. We also randomly select κ of the non-edge pixels, with κ in $(0,1]$ and $\kappa \leq \rho$. The non-edge pixels usually have less reliability in the local motion estimation searching process but can be used to improve the uniformity of the sampling scheme and the final dense motion field estimation. Assume $S = \{(i,j)\}$ is the set of chosen pixels for sparse motion estimation. We first calculate the Delaunay triangulation of the point-set S. The triangulation result is represented as the graph G whose edges define the neighborhood relation $\mathcal{N}$ of the chosen points, in the linear programming formulation: if there is an edge in G between node (i,j) and (k,l), then we have $[(i,j),(k,l)] \in \mathcal{N}$. In the actual implementation we only select either $[(i,j),(k,l)]$ or $[(k,l),(i,j)]$ in $\mathcal{N}$.

We have defined the neighbor set $\mathcal{N}$ and we need further to define the motion basic set $\mathcal{B}_{i,j}$ for site (i,j). We confine the matching to a rectangular window centered at site (i,j) in the reference image which corresponds to the motion candidate set $W = [\Delta_{x\min}, \Delta_{x\max}] \times [\Delta_{y\min}, \Delta_{y\max}]$. It is not difficult to show that the best motion basis set for each site can be selected as the motions corresponding to the vertices of the lower convex hull of the cost surface of all the motion candidates in window W. In this paper, we simply take the cost of assigning motion (m,n) to site (i,j) to be the *color consistency* between reference and matching blocks:

$$C_{i,j,m,n} = \frac{\sum_{(s,t)\in O_{i,j}} |I_r(s,t) - I_m(s+m,t+n)|}{L \cdot (\sigma_r^2 + 10^{-4})^{0.5} (\sigma_m^2 + 10^{-4})^{0.5}}$$

where I_r and I_m are the normalized reference and matching gray level images, respectively; $O_{i,j} = [i-\tau, i+\tau] \times [j-\tau, j+\tau]$, where τ is 1..4 in this paper; $L = (2\tau+1)^2$; $\sigma_r^2 = \frac{1}{L-1}\sum_{(s,t)\in O_{i,j}} (I_r(s,t) - m_r)^2$ and $\sigma_m^2 = \frac{1}{L-1}\sum_{(s,t)\in O_{i,j}} (I_m(s+m,t+n) - m_m)^2$ are

the non-deviation estimation of the variance of the image blocks; m_r and m_m are the mean value for the reference and matching image block.

In our scheme, we would like the smoothing terms $\lambda_{i,j,k,l}$ and $\mu_{i,j,k,l}$ adapted to the distance of neighboring sites: $\lambda_{i,j,k,l} = \lambda_0 w(d_{i,j,k,l})$ and $\mu_{i,j,k,l} = \mu_0 w(d_{i,j,k,l})$ where $d_{i,j,k,l} = ((i-k)^2 + (j-l)^2)^{1/2}$ and $w(\cdot)$ is a decreasing function in the domain $[0,1]$, with λ_0 and μ_0 two constant coefficients. The typical values of λ_0 and μ_0 are 0.01 and 0.02 respectively. C_0 is in $[0.5, 0.7]$. In this paper, $w(\cdot)$ is selected as a simple staircase function equal to 0 if the distance between two neighbors is greater than some threshold and is otherwise 1.

3.2 Dense Refinement Based on PDE Method

Based on the above linear programming method, we obtain a sparse reconstruction for the motion in the x and y directions, as well as the occlusion map. The sparse reconstruction needs further refinement to derive a dense estimation at a resolution equal to that of the original images. The occlusion map can easily be extended to a dense map by a simple interpolation scheme. In this section we give a scheme to refine the x and y motion field into a dense motion field.

We use the interpolated sparse x-, y-motion field as the initial motion, and apply a PDE-based method to refine the estimation in a force field determined by the consistency function in the two images. To begin with, we interpolate the sparse x-motion set at points dx_{x_i,y_i} so as to generate a dense set $h(x,y)$. A straightforward approach for achieving this is to solve a constrained Laplace's equation:

$$\begin{cases} \nabla^2 h = 0 \\ h(x_i, y_i, t) \equiv dx_{x_i,y_i},\ t > 0, i = 1...M \end{cases}$$

for h. To do so, we iterate using an artificial time variable t, and constrain a solution at any time to pass through the sparse mesh points. Dense x-motion $h(x,y)$ is analogue to the steady state for isotropic heat transmission with constant temperatures at the mesh nodes. In the discretized domain above, we have converted a non-regular grid into a much easier regular-grid system. In the rest of the analysis, we will work on this regular grid. This interpolation scheme iterates the standard heat diffusion equation. y-motion is interpolated with a similar scheme and the dense y-motion is denoted $v(x,y)$.

We further formulate the following variational problem to derive smooth motion fields p and q (x and y motions) from the initial interpolated h and v:

$$\begin{gathered} \{\widehat{p}, \widehat{q}\} = \min_{p,q} \mathcal{J} = \int_x \int_y [\eta \cdot G(x,y,p,q) \\ + \frac{(1-\alpha(x,y))}{2} \|\nabla p(x,y)\|^2 + \frac{(1-\beta(x,y))}{2} \|\nabla q(x,y)\|^2 \\ + \frac{\alpha(x,y)}{2} \|\nabla p(x,y)\| + \frac{\beta(x,y)}{2} \|\nabla q(x,y)\|] dxdy, \\ \eta = \text{const}; G = \text{consistency function} \end{gathered} \tag{1}$$

with constraints

$$\begin{gathered} \frac{p_y}{\sqrt{p_x^2+p_y^2}} = \sin[\theta(x,y)], \frac{p_x}{\sqrt{p_x^2+p_y^2}} = \cos[\theta(x,y)], \\ \frac{q_y}{\sqrt{q_x^2+q_y^2}} = \sin[\varphi(x,y)], \frac{q_x}{\sqrt{q_x^2+q_y^2}} = \cos[\varphi(x,y)], \\ \alpha = f^{(x)}(\|\nabla h^\sigma(x,y)\|), \beta = f^{(y)}(\|\nabla v^\sigma(x,y)\|), \end{gathered} \tag{2}$$

where the $\{\sin, \cos\}$ constraints signify that p and q are surfaces. We preserve smooth motions by letting constant function $\alpha(x,y)$ or $\beta(x,y)$ be small if the initial interpolating motion h or v is smooth. In that case, we would like the second and third terms in $\mathcal{J}$ to dominate and color consistency to guide further smoothing. However, for non-smooth motions we would like the fourth and fifth, curvature-producing, terms to be more important and therefore $\alpha(x,y)$ or $\beta(x,y)$ should be larger. Coefficient η is a constant that controls the influence of the energy term based on the consistency function. $f^{(\cdot)}(x) = \max((\frac{x - x_{\min}}{x_{\max} - m_{\min}})^{0.2}, 0.95)$, where $x_{\min}$ and $x_{\max}$ are the minimum and maximum gradients in the domain of image pixels.

So we use the initial interpolating motion h and v to set a general smoothness level by defining positive monotonic-increasing functions $f^{(x)}(\cdot)$ and $f^{(y)}(\cdot)$ both with range $[0,1]$, so that α and β go from low to high as smoothness decreases. To simplify the variational equation below, we further assume that the motion field can be represented as piecewise planar surfaces. Thus the gradient of the x-motion field and y-motion field are piecewise constant. Based on the definitions of α and β, we have $\partial\alpha/\partial x \simeq 0, \partial\alpha/\partial y \simeq 0, \partial\beta/\partial x \simeq 0, \partial\beta/\partial y \simeq 0$ almost everywhere. A simplifying assumption, then, is that the partial derivatives of the α and β vanish. Based on this assumption, the resulting PDE below becomes much simpler, crucially for an iterative algorithm, at the expense of slightly sacrificing the edge preserving property.

We also use the initial normal vectors ∇h and ∇v to set the normal direction for a solution, ∇p and ∇q, by constraining p and q via the first four conditions in (2). This makes the algorithm fast. To fix the normal direction, we take $\theta(x,y) = \arctan(h_y^\sigma / h_x^\sigma)$, with h^σ the initial dense x-motion h, Gaussian-smoothed at scale σ, and similarly $\varphi(x,y) = \arctan(v_y^\sigma / v_x^\sigma)$. $G(x,y,p,q)$ is a real function defining color consistency of the point (x,y) with motion (p,q): $G(x,y,p,q)$ is an interpolated version of $C_{i,j,m,n}$. $G(x,y,p,q)$ is minimized if the color block centered at (x,y) in the reference image is consistent with the color block at $(x+p, y+q)$ in the matching image.

The Euler-Lagrange equations for eq. (1) are given by the variational derivatives $\delta\mathcal{J}/\delta p = 0, \delta\mathcal{J}/\delta q = 0$, and applying the vanishing assumptions for the partial derivatives of α and β we have

$$\begin{gathered} \eta G_p(x,y,p,q) - (1-\alpha)(p_{xx} + p_{yy}) \\ -\alpha \frac{p_{xx}p_y^2 + p_{yy}p_x^2 - 2p_{xy}p_x p_y}{2(p_x^2+p_y^2)^{3/2}} = 0, \\ \eta G_q(x,y,p,q) - (1-\beta)(q_{xx} + q_{yy}) \\ -\beta \frac{q_{xx}q_y^2 + q_{yy}q_x^2 - 2q_{xy}q_x q_y}{2(q_x^2+q_y^2)^{3/2}} = 0 \end{gathered} \tag{3}$$

The third term in both equations— the curvature for equi-value contour curves for fields p and q— smooths only along contour tangents (when the terms contribute enough, i.e., at edges), not isotropically like the second term.

Substituting eq. (2) into (3), by introducing an artificial time variable t and adding a small positive value ε to avoid a zero denominator, the partial differential equation solution

of the system can be written

$$p_t = p_{xx}(1-\alpha+\alpha\frac{\sin^2\theta}{2(p_x^2+p_y^2)^{1/2}+\varepsilon})$$
$$+p_{yy}(1-\alpha+\alpha\frac{\cos^2\theta}{2(p_x^2+p_y^2)^{1/2}+\varepsilon})$$
$$-p_{xy}\frac{\alpha\sin\theta\cos\theta}{(p_x^2+p_y^2)^{1/2}+\varepsilon}-\eta G_p(x,y,p,q),$$
$$q_t = q_{xx}(1-\beta+\beta\frac{\sin^2\varphi}{2(q_x^2+q_y^2)^{1/2}+\varepsilon})$$
$$+q_{yy}(1-\beta+\beta\frac{\cos^2\varphi}{2(q_x^2+q_y^2)^{1/2}+\varepsilon})$$
$$-q_{xy}\frac{\beta\sin\varphi\cos\varphi}{(q_x^2+q_y^2)^{1/2}+\varepsilon}-\eta G_q(x,y,p,q)$$

In summary, the x- and y-motion are initialized to h and v respectively, the dense interpolation of the sparse motion. As time t increases, the above PDE approaches a standard heat equation with a penalty term given by the consistency in two images in smooth motion regions. The diffusion method approaches a mean curvature motion solution at rapidly changing areas such as edges or corners. A finite difference method is used in the discretization of the PDEs. The coupled PDEs are iterated alternatively between the x- and y-motion fields until the result converges.

4 Experiment Results

In this section, we present the results of the linear programming based motion estimation method in the application of motion estimation and stereo problems. In all the following experiments, we use only the gray scale image for feature extraction and matching. If the image is a color image, it will be first converted to a gray level image before further processing. The gray level image is also normalized such that the gray levels are in [0,1].

First, we compare the proposed LP based method with the graph-cut method, the state of art benchmark global optimization scheme. To simplify the comparison, we ignore occlusion in these experiments for both methods. As well, we use the same cost function and topology for both methods. About 800 random points are selected in a rectangular area and weighting parameters are tuned such that the best performance in the sense of mean absolute error is attained for both schemes respectively. Fig. 1 and Fig. 2 show test images used and their ground truth motion field and the best-performance interpolated sparse motion estimation results for two different experiments. Mean absolute error for the selected points is compared in Fig. 3. For these tests, the LP based method has substantially better performance both visually and in the sense of mean absolute error.

Fig. 4 illustrates the sparse linear programming edge-feature points based matching result and dense motion estimation for image `Table`. A rectangular region has been selected as the region of interest manually. About 2000 edge-points and supporting points are detected in the region of interest. The local searching region is $[-20, 20]$ in both x and

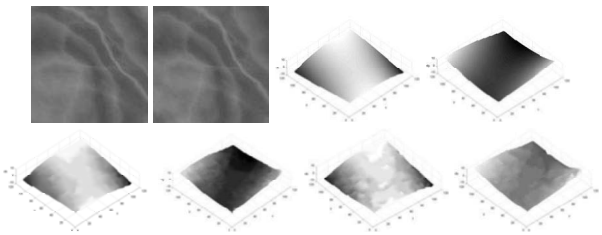

Figure 1: Top left two: reference and matching images. Top right two: ground truth x and y motion field. Bottom left two: interpolated LP motion estimation. Bottom right two: interpolated graph cut motion estimation.

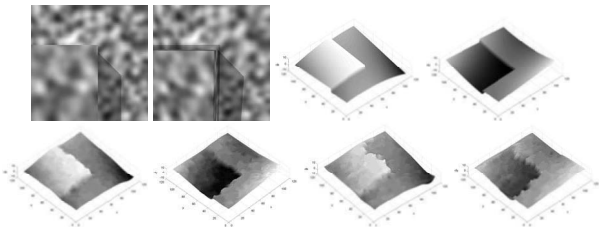

Figure 2: Top left two: reference and matching images. Top right two: ground truth x and y motion field. Bottom left two: interpolated LP motion estimation. Bottom right two: interpolated graph cut motion estimation.

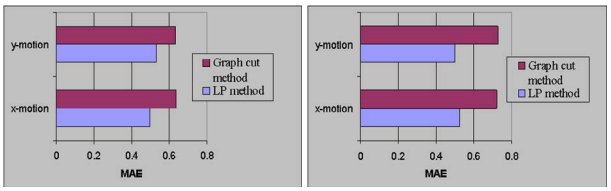

Figure 3: MAE comparison: Left, for Fig. 1 and Right, for Fig. 2. In each comparison, y-motion error is at the top and x-motion is the bottom set of bars. The darker bars, on top, show error for the Graph-cut method, while the lighter, bottom bars are for the proposed LP based method.

y directions. The basis sets for each site are selected based on the scheme proposed in § 3.1. We calculate the dense motion field in Fig. 4 based on the proposed two-phase scheme. To visualize the dense motion estimation result, we plot the matching result on *regular* grids based on the dense motion estimation, in Fig. 4. Fig. 5 shows the dense motion estimation result based on the graph-cut scheme in [12]. In this experiment, the graph-cut method uses 3 iterations and takes about 15 minutes to finish with a Pentium-4 2.6GHz Linux PC. The proposed two-stage scheme takes about 5 minutes. More experimental results, using image `Toy_house`, are shown in Fig. 6.

In the above experiments, there is little occlusion involved. Figs. 7, 8, and 9 illustrate experiment results for estimating motion and the occlusion map simultaneously with the proposed method for image `Mouse`. In this experiment, camera motion and object motion are both involved. Another challenge of the experiment is that the mouse and mouse pad contain large areas without texture. The scaled

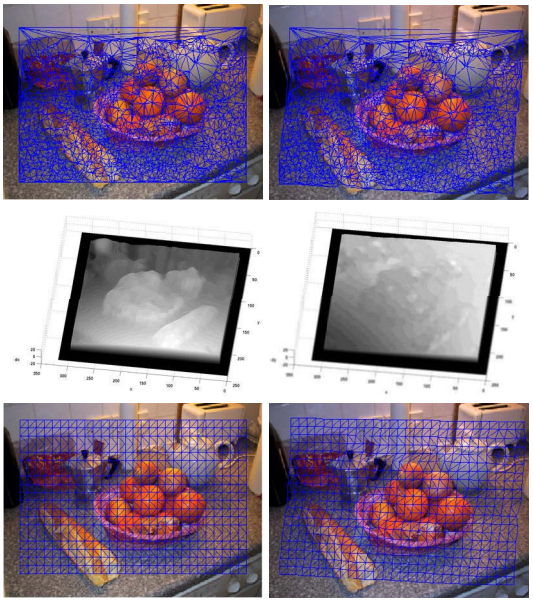

Figure 4: `Table`. Top left: reference image and mesh. Top right: matching image and matching mesh based on sparse LP method. Middle: dense x motion (left); dense y motion (right). Bottom: regular mesh matching based on dense motion field.

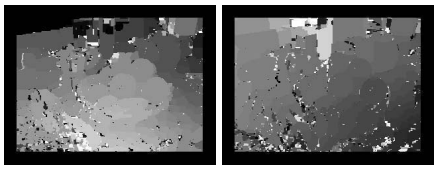

Figure 5: Dense x-motion (left) and y-motion (right) estimation based on the graph-cut method.

motion vector plot is shown in Fig. 7. A threshold of 0.5 is used to obtain the occlusion map in the experiments. In Fig. 7, the occlusion map is shown aligned with the reference image by replacing the red channel of the reference image with the occlusion map while keeping intact the green and blue channels. Fig. 8 shows the sparse matching result based on the proposed linear programming scheme and Fig. 9 shows the dense motion field based on the proposed two-phase motion estimation scheme.

We also apply the proposed scheme to the stereo problem with large occlusions. Fig. 10 shows the stereo pair for the image `Map` with the motion vector plot and occlusion map in a polygonal window region of interest. Fig. 11 shows the dense disparity map and its texture-mapped image in the region of interest. Fig. 12 shows the dense disparity estimation without occlusion inference. Comparing the texture mapped images of Figs. 11 and 12, we can see that *the occlusion inference improves the motion at the boundary of occlusion*.

In experiments, we found that limitations of the method have to do with sharp boundaries and with noise. Because of the motion smoothing term, occasionally at a sharp object boundary motion estimation results near the boundary may include influence from a different object, resulting in a minor error. As well, if the image is too noisy, distinguishing the occlusion map becomes difficult because error in the motion estimation and error in identifying occlusions becomes too close.

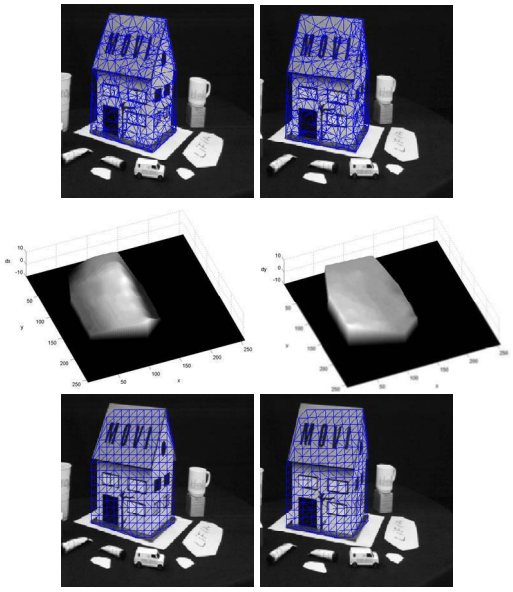

Figure 6: `Toy_house`. Top left: reference image and mesh. Top right: matching image and matching mesh based on sparse LP method. Middle: dense x motion (left). dense y motion (right). Bottom: regular mesh matching based on dense motion field.

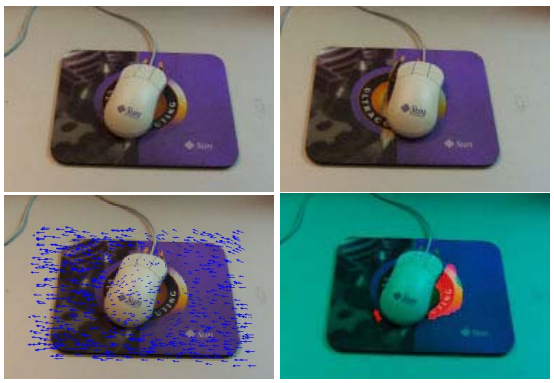

Figure 7: `Mouse`. Upper images: reference and matching images. Lower left: scaled motion vectors. Lower right: occlusion map shown in red.

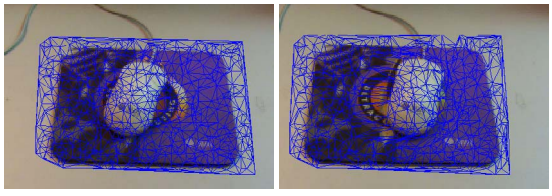

Figure 8: Sparse matching based on LP method.

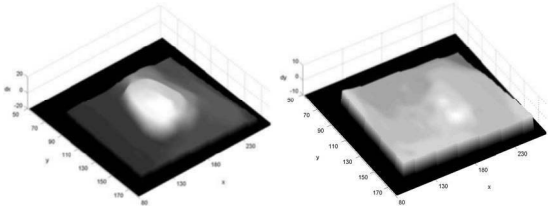

Figure 9: Dense x-motion (left) and y-motion (right) based on the proposed two-phase method.

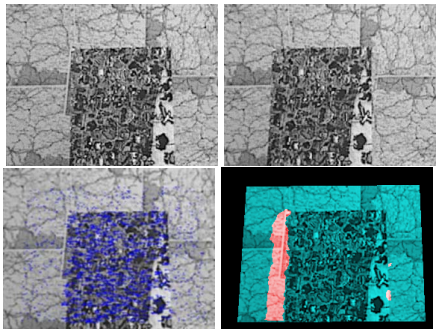

Figure 10: `Map`. Upper images: left and right view. Lower left: scaled motion vectors. Lower right: occlusion map shown in red.

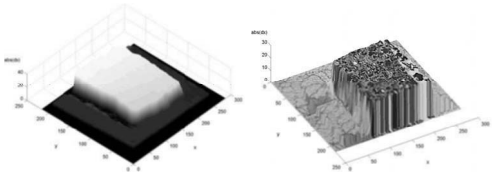

Figure 11: Dense disparity map, and texture-mapped figure.

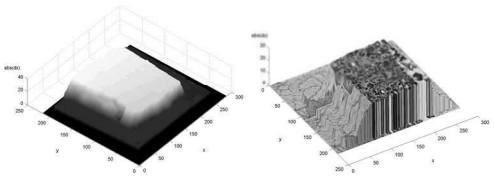

Figure 12: Dense disparity map *without* occlusion inference; texture-mapped figure.

5 Conclusion

In this paper, we propose a new linear programming based method to estimate large scale motion in both x and y directions in two images. The proposed method estimates the occlusion map and motion field at the same time. To solve the still relatively high-complexity problem of pure linear programming, we present a two-phase scheme to estimate a dense motion field. In comparison to graph-cut or belief propagation for the general motion estimation problem, the proposed LP method has a different optimization mechanism. We uniquely represent the search space via a basis set, with dimension much smaller than the full candidate set used for the graph-cut or belief propagation methods. Crucially, this dimension reduction does not affect the solution space — it still equals the space spanned by the whole candidate set. We thus remove some correlation between neighbor candidates, high for motion estimation, increasing speed without degrading performance. Also, we inherently generate a float solution, much more straightforwardly than the subpixel graph-cut and belief propagation method. Applying the LP formulation iteratively with the search range shrinking during each iteration will be a future study direction.

References

[1] B.K.P. Horn and B.G. Schunck, Determining optical flow, AI Journal, Vol.17, p. 185-203, 1981.

[2] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In IJCAI, p. 674-679, 1981

[3] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, Performance of optical flow techniques, IJCV, Vol.12, p. 43-77, 1994

[4] H. Liu, R. Chellappa, and A. Rosenfeld, Accurate dense optical flow estimation using adaptive structure tensors and a parametric model, IEEE Trans. Image Proc., Vol.12, p. 1170-1180, 2003

[5] Y. Boykov, O. Veksler, and R. Zabih, Fast approximate energy minimization via graph cuts, PAMI, Vol.23, p. 1222-1239, 2001.

[6] V. Kolmogorov and R. Zabih, Multi-camera scene reconstruction via graph cuts. ECCV, p. III: 82 ff., 2002.

[7] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts?, ECCV, p. III: 65 ff., 2002.

[8] S. Roy and I.J. Cox, A maximum flow formulation of the N-camera stereo correspondence problem. ICCV, p. 429-499, 1998.

[9] H. Ishikawa, Global Optimization using Embedded Graphs, Ph.D. Dissertation. May 2000, NYU.

[10] J. Sun, N.-N. Zheng and H.-Y. Shum, Stereo matching using belief propagation, PAMI, Vol. 25, p. 787-800, 2003.

[11] C.F. Olson, Maximum-likelihood image matching, PAMI, Vol. 24, p. 853-857, 2002.

[12] V. Kolmogorov and R. Zabih, Computing visual correspondence with occlusions using graph cuts, ICCV, p. II. 508-515, 2001.

[13] S.T. Barnard, Stochastic stereo matching over scale, IJCV, Vol.3, p. 17–32, 1989.

[14] J. Maciel and J.P. Costeira, A global solution to sparse correspondence problems, PAMI, Vol.25, p. 187-199, 2003.

[15] M. Ben-Ezra, S. Peleg, and M. Werman, Real-time motion analysis with linear-programming, ICCV, p. II.703-709, 1999.

[16] M. Ben-Ezra, S. Peleg and M. Werman, Model based pose estimator using linear programming, ECCV, p. 267-281, 2000.

[17] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, Approximation algorithms for the metric labeling problem via a new linear programming formulation, Symp. on Discrete Algs. p. 109-118, 2001.

[18] V. Chvátal. Linear Programming, W. H. Freeman and Co., New York, 1983.