21^{st} *Computer Vision Winter Workshop*
Luka Čehovin, Rok Mandeljc, Vitomir Štruc (eds.)
Rimske Toplice, Slovenia, February 3–5, 2016

Solving Dense Image Matching in Real-Time using Discrete-Continuous Optimization

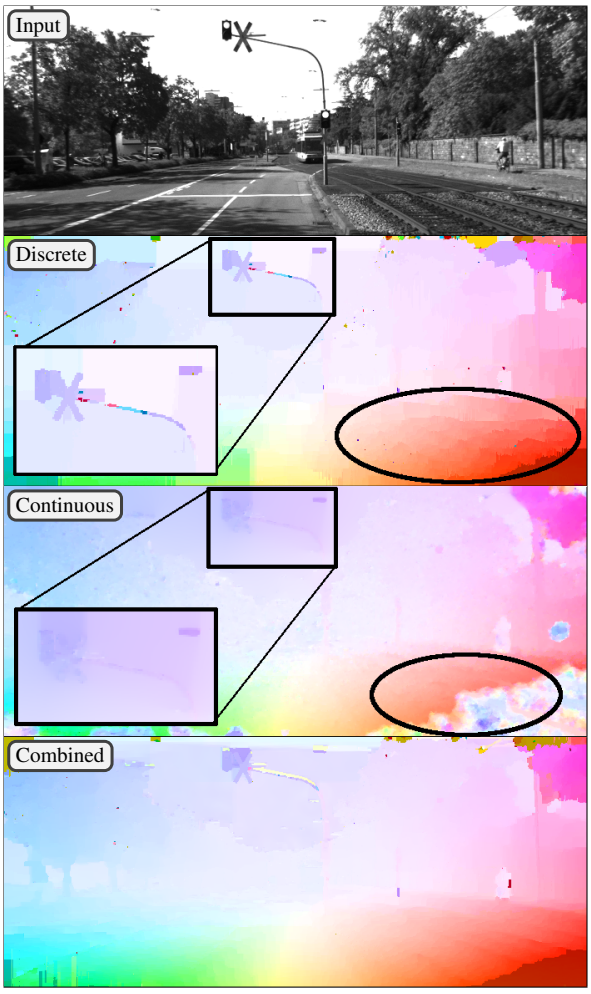
Alexander Shekhovtsov, Christian Reinbacher, Gottfried Graber and Thomas Pock
Institute for Computer Graphics and Vision, Graz University of Technology
{shekhovtsov,reinbacher,graber,pock}@icg.tugraz.at

Abstract. *Dense image matching is a fundamental low-level problem in Computer Vision, which has received tremendous attention from both discrete and continuous optimization communities. The goal of this paper is to combine the advantages of discrete and continuous optimization in a coherent framework. We devise a model based on energy minimization, to be optimized by both discrete and continuous algorithms in a consistent way. In the discrete setting, we propose a novel optimization algorithm that can be massively parallelized. In the continuous setting we tackle the problem of non-convex regularizers by a formulation based on differences of convex functions. The resulting hybrid discrete-continuous algorithm can be efficiently accelerated by modern GPUs and we demonstrate its real-time performance for the applications of dense stereo matching and optical flow.*

1. Introduction

The *dense image matching* problem is one of the most basic problems in computer vision: The goal is to find matching pixels in two (or more) images. The applications include stereo, optical flow, medical image registration, face recognition [1], *etc*. Since the matching problem is inherently ill-posed, typically *optimization* is involved in solving it. We can distinguish two fundamentally different approaches: discrete and continuous optimization. Whereas discrete approaches (see [14] for a recent comparison) assign a distinct label to each output pixel, continuous approaches try to solve for a *function* using the calculus of variations [6, 8, 21]. Both approaches have received enormous attention, and there exist state-of-the-art algorithms in both camps: continuous [23, 24, 28] and discrete [18, 30]. Due to the specific mathematical tools available to solve the problems (discrete combinatorial optimization vs. continuous calculus of variations), both approaches have distinct advantages and disadvantages.

In this paper, we argue that on a fundamental level the advantages and disadvantages of discrete and continuous optimization for dense matching problems are *complementary* as summarized in Figure 1. The previous work combining discrete and continuous optimization primarily used discrete optimization to fuse (find the optimal



	data term	Large motion	Parallelization
Discrete	Arbitrary (sampled)	Easy	Difficult
Continuous	Convex (linearized)	Difficult	Easy

Figure 1: Optical flow problem solved by a purely discrete method, a purely continuous method and the combined method. All methods are as described in this paper, they use the same data term and are run until convergence here. In the discrete solution we can see small scale details and sharp motion boundaries but also discretization artifacts. The continuous solution exhibits sub-pixel accuracy (smoothness), but lacks small details and has difficulties with large motions. The combined solution delivers smooth flow fields while retaining many small scale details.

crossover) of candidate continuous proposals, *e.g*. [36, 30] (stereo) and [25] (flow). The latter additionally performs local continuous optimization of the so-found solution. Many works also alternate between continuous and discrete optimizations, addressing a Mumford-Shah-like model, *e.g*., [5]. Similarly to [25] we introduce a continuous energy which is optimized using a combined method. However, we work with a full (non-local) discretization of this model and propose new parallel optimization methods.

The basic difference in discrete and continuous approaches lies in the handling of the data term. The data term is a measure how well the solution (*i.e*. value of a pixel) fits the underlying measurement (*i.e*. input images). In the discrete setting, the solution takes discrete labels, and hence the number of labels is finite. Typically the data cost is precomputed for all possible labels. The discrete optimization then uses the data cost to find the optimal label for each pixel according to a suitable model in an energy minimization framework. We point out that due to the *sampling* in both label space and spatial domain, the discrete algorithm has access to the full information at every step. *I.e*. it deals with a *global optimization* model and in some lucky cases can find a globally optimal solution to it or provide an approximation ratio or partial optimality guarantees [27].

In the continuous setting, the solution is a continuous function. This means it is not possible to precompute the data cost; an infinite number of solutions would require infinite amount of memory. More importantly, the data cost is a non-convex function stemming from the similarity measure between the images. In order to make the optimization problem tractable, a popular approach is the linearization of the data cost. However, this introduces a range of new problems, namely the inability to deal with large motions due to the fact that the linearization is valid only in a small neighborhood around the linearization point. Most continuous methods relying on linearization therefore use a coarse-to-fine framework in an attempt to overcome this problem [4]. One exception is a recent work [16], which can handle piece-wise linear data terms and truncated TV regularization.

Our goal in this paper is to combine the advantages of both approaches, as well as real-time performance, which imposes tough constraints on both methods resulting in a number of challenges:

Challenges The discrete optimization method needs to be highly parallel and able to *couple* the noisy / ambiguous data over large areas. The continuous energy should be a refinement of the discrete energy so that we can evaluate the two-phase optimization in terms of a single energy function. The continuous method needs to handle robust (truncated) regularization terms.

Contribution Towards the posed challenges, we propose: i) a new method for the discrete problem, working in the dual (*i.e*. making equivalent changes of the data cost volume), in parallel on multiple chains; ii) a continuous optimization method, reducing non-convex regularizers to a primal-dual method with non-linear operators [31]; iii) an efficient implementation of both methods on GPU and proof of concept experiments showing advantages of the combined approach.

2. Method

In this section we will describe our two-step approach to the dense image matching problem. To combine the previously discussed advantages of discrete and continuous optimization methods it is essential to minimize the same energy in both optimization methods. Starting from a continuous energy formulation in § 2.1, we first show how to discretize the energy in § 2.2 and subsequently minimize it using a novel discrete parallel block coordinate descent, described in § 2.3. The output of this algorithm will be the input to a refinement method which is posed as a continuous optimization problem, solved by a non-linear primal-dual algorithm described in § 2.4.

2.1. Model

Let us formally define the dense image matching problem to be addressed by the discrete-continuous optimization approach. In both formulations we consider that the image domain is a discrete *set of pixels* $\mathcal{V}$. The continuous formulation has continuous ranged variables $u = (u_i^k \in \mathbb{R} \mid k = 1, \dots d,\ i \in \mathcal{V})$, where $d = 1, 2$ for stereo / flow, respectively. The matching problem is formulated as

$$\min_{u \in U} \Big[E(u) = D(u) + R(Au) \Big], \tag{1}$$

where $U = \mathbb{R}^{d \times \mathcal{V}}$; D is the *data term* and $R(Au)$ is a *regularizer* (A is a linear operator explained below). The discrete formulation will quantize variable ranges.

Data Term We assume $D(u) = \sum_{i \in \mathcal{V}} D_i(u_i)$, where $D_i \colon \mathbb{R}^d \to \mathbb{R}$ encodes the deviation of u_i from some underlying measurement. A usual choice for dense image matching are robust filters like Census Transform or Normalized Cross Correlation, computed on a small window around a pixel. This data term is non-convex in u and piecewise linear. In the discrete setting, the data term is sampled at discrete locations, in the continuous setting, the data term is convexified by linearizing or approximating it around the current solution. The details will be described in the respective sections.

Regularization Term The regularizer encodes properties of the solution of the energy minimization like local smoothness or preservation of sharp edges. The choice of this term is crucial in practice, since the data term may be unreliable or uninformative in large areas of dense matching problems. We assume

$$R(Au) = \sum_{ij \in \mathcal{E}} \omega_{ij} \sum_{k=1}^{d} r((Au^k)_{ij}), \tag{2}$$

where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of *edges*, *i.e*., pairs of neighboring pixels; linear operator $A \colon \mathbb{R}^{\mathcal{V}} \to \mathbb{R}^{\mathcal{E}} \colon u^k \mapsto$

$(u_i^k - u_j^k \in \mathbb{R} \,|\, \forall ij \in \mathcal{E})$ essentially computes gradients along the edges in $\mathcal{E}$ for the solution dimension k; the gradients are penalized by the *penalty function* $r\colon \mathbb{R} \to \mathbb{R}$ and ω_{ij} are image dependent per-edge strength weights, reducing the penalty around sharp edges. Our particular choice for the penalty function r is depicted in Fig. 2. We chose to use a truncated norm which has shown to be robust against noise that one typically encounters in dense matching problems. It generalizes truncated Total Variation in the continuous setting. In the discrete setting it generalizes the $P1$-$P2$ penalty model [11], Potts model and the truncated linear model.

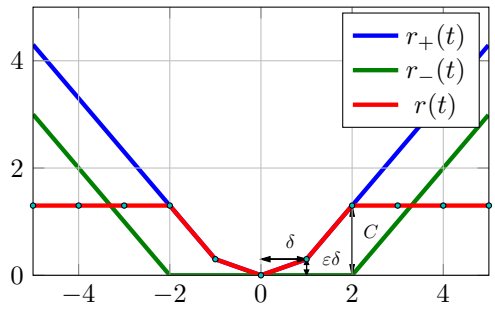


Figure 2: Regularizer function r. In our continuous optimization method it is decomposed into a difference of convex functions $r_+ - r_-$. For the discrete optimization it is sampled at label locations depicted as dots.

2.2. Discrete Formulation

In the discrete representation we will use the following formalism. To a continuous variable u_i we associate a discrete variable $x_i \in \mathcal{L}$. The discrete label space $\mathcal{L}$ can be chosen to our convenience as long as it has the desired number of elements, denoted K. We let $\mathcal{L}$ to be vectors in $\{0,1\}^K$ with exactly one component equal 1 (the 1-hot encoding of natural numbers from 1 to K). For $f_i \in \mathbb{R}^K$ we denote $f_i(x_i) = \langle f_i, x_i \rangle = f_i^\mathsf{T} x_i$ and for $f_{ij} \in \mathbb{R}^{K \times K}$ we denote $f_{ij}(x_i, x_j) = x_i^\mathsf{T} f_{ij} x_j$. Let $f = (f_w \,|\, w \in \mathcal{V} \cup \mathcal{E})$ denote the energy *cost vector*. The *energy function* corresponding to the cost vector f is given by

$$f(x) = \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{ij \in \mathcal{E}} f_{ij}(x_i, x_j). \tag{3}$$

Whenever we need to refer to f as a function and not as the cost vector, we will always use the argument notation, *e.g.* $f(x) \geq g(x)$ is different from $f \geq g$.

Energy function f that can be written as $\sum_i f_i(x_i) = \langle f, x \rangle$ is called *modular*, *separable* or *linear*. Formally, all components f_{ij} of f are identically zero. If f_{ij} is non-zero only for a subgraph of $(\mathcal{V}, \mathcal{E})$ which is a set of chains, we say that f is a *chain*.

The *discrete energy minimization* problem is defined as

$$\min_{x \in \mathcal{L}^\mathcal{V}} f(x). \tag{4}$$

Stereo We discretize a range of disparities and let $u(x) \in \mathbb{R}^\mathcal{V}$ denote the continuous solution corresponding to the *labeling* x. We set $f_i(x_i) = D_i(u(x_i))$ and $f_{ij}(x_i, x_j) = \omega_{ij} r((Au(x))_{ij})$.

Flow Discretization of the flow is somewhat more challenging. Since u_i is a 2D vector, assuming large displacements, discretizing all combinations is not tractable. Instead, components u_i^1 and u_i^2 can be represented as separate discrete variables x_{i^1}, x_{i^2} , where (i^1, i^2) is a pair of nodes duplicating i, leading to the *decomposed* formulation [26]. To retain the pairwise energy form (3), this approach assigns the data terms $D_i(u_i)$ to a pairwise cost $f_{i^1 i^2}(x_{i^1}, x_{i^2})$ and the regularization is imposed on each layer of variables $(x_{i^1} \,|\, i \in \mathcal{V})$ and $(x_{i^2} \,|\, i \in \mathcal{V})$ separately. To this end, we tested a yet simpler representation, in which we assign optimistic data costs, given by

$$f_{i^1}(x_{i^1}) = \min_{x_{i^2}} D_i(x_{i^1}, x_{i^2}), \tag{5a}$$
$$f_{i^2}(x_{i^2}) = \min_{x_{i^1}} D_i(x_{i^1}, x_{i^2}), \tag{5b}$$

where $D_i(x_{i^1}, x_{i^2})$ is the discretized data cost, and regularize in each layer individually. This makes the two layers fully decouple into, essentially, a two independent stereo-like problems. At the same time, the coupled scheme [26], passing messages between the two layers, differs merely in recomputing (5) for a reparametrized data costs in a loop. Our simplification then is not a principled limitation but an intermediate step.

2.3. Discrete Optimization

In this section we give an overview of a new method under development addressing problem (4) through its LP-relaxation dual. In real-time applications like stereo and flow there seem to be a demand in methods performing fast approximate discrete optimization, preferably well-parallelizable. It has motivated a significant research. The challenge may sound as "*best solution in a limited time budget*".

Well-performing methods, from local to global, range from cost volume filtering [12], semi-global matching (SGM) [11] (has been implemented in GPU and FPGA [2]), dynamic programming on spanning trees adjusting the cost volume [3] and more-global matching (MGM) [10] to the sequential dual block coordinates methods, such as TRW-S [15]. Despite being called sequential, TRW-S exhibits a fair amount of parallelism in its computation dependency graph, which is exploited in the parallel GPU/FPGA implementations [7, 13]. At the same time SGM has been interpreted [9] as a single step of parallel TRW algorithm [32] developed for solving the dual. MGM goes further in this direction, resembling even more the structure of a dual solver: it combines together more messages but in a heuristic fashion and introducing more computation dependencies, in fact similar to TRW-S. It appears that all these approaches go somehow in the direction of a fast processing of the dual.

We propose a new dual update scheme, which: i) is a monotonous block-coordinate ascent; ii) performs as good

as TRW-S for an equal number of iterations while having a comparable iteration cost; and iii) offers more parallelism, better mapping to current massively parallel compute architectures. Thus it bridges the gap between highly parallel heuristics and the best "sequential" dual methods without compromising on the speed and performance.

On a higher level, the method is most easily presented in the dual decomposition framework. For clarity, let us consider a decomposition into two subproblems only (horizontal and vertical chains). Consider minimizing the energy function $E(x)$ that separates as

$$E(x) = f(x) + g(x), \tag{6}$$

where $f,\ g\colon \mathcal{L}^{\mathcal{V}} \to \mathbb{R}$ are chains.

Primal Majorize-Minimize Even before introducing the dual, we can propose applying the majorize-minimize method (a well-known optimization technique) to the primal problem in the form (6). It is instructive for the subsequent presentation of the dual method and has an intriguing connection to it, which we do not yet fully understand.

Definition 2.1. A modular function $\bar{f}$ is a *majorant* (upper bound) of f if $(\forall x)\ \ \bar{f}(x) \geq f(x)$, symbolically $\bar{f} \succeq f$. A modular minorant $\underline{f}$ of f is defined similarly.[1]

Noting that minimizing a chain function plus a modular function is easy, one could straightforwardly propose Algorithm 1, which alternates between majorizing one of f or g by a modular function and minimizing the resulting chain problem $\bar{f} + g$ (resp. $f + \bar{g}$). We are not aware of this approach being evaluated before. Somewhat novel, the sum of two chain functions is employed rather than, say, difference of submodular [19], but the principle is the same. To ensure monotonicity of the algorithm we need to pick a majorant $\bar{f}$ of f which is exact in the current primal solution x^k as in Line 1. Then $f(x^{k+1}) + g(x^{k+1}) \leq \bar{f}(x^{k+1}) + g(x^{k+1}) \leq \bar{f}(x^k) + g(x^k) = f(x^k) + g(x^k)$. Steps 3-4 are completely similar. Algorithm 1 has the following properties:

- primal monotonous;
- parallel, since, *e.g.*, $\min_x(\bar{f} + g)(x)$ decouples over all vertical chains;
- uses more information about subproblem f than just the optimal solution (as in most primal block-coordinate schemes: ICM, alternating lines, *etc.*).

The performance of this method highly depends on the strategy of choosing majorants. This will be also the main question to address in the dual setting.

Dual Decomposition Minimization of (6) can be written as

$$\min_{x^1 = x^2} f(x^1) + g(x^2). \tag{7}$$

Introducing a vector of Lagrange multipliers $\lambda \in \mathbb{R}^{\mathcal{L} \times \mathcal{V}}$ for the constraint $x^1 = x^2$, we get the Lagrange dual problem:

$$\max_{\lambda} \Big[\underbrace{\min_x \big(f(x) + \langle \lambda, x \rangle\big)}_{D^1(\lambda)} + \underbrace{\min_x \big(g(x) - \langle \lambda, x \rangle\big)}_{D^2(\lambda)} \Big]. \tag{8}$$

The so-called *slave* problems $D^1(\lambda)$ and $D^2(\lambda)$ have the form of minimizing an energy function with a data cost modified by λ. The goal of the *master problem* (8) is to balance the data cost between the slave problems such that their solutions agree. The slave problems are minima of finitely many functions linear in λ, the objective of the master problem (8) $D(\lambda) = D^1(\lambda) + D^2(\lambda)$ is thus a concave piece-wise linear function. Problem (8) is a concave maximization. However, since x was taking values in a discrete space, there is only a weak duality: (7) $\geq$ (8). It is known that (8) can be written as a linear program (LP), which is as difficult in terms of computation complexity as a general LP [22].

Dual Minorize-Maximize In the dual, which is a maximization problem, we will speak of a minorize-maximize method. The setting is similar to the primal. We can *efficiently* maximize D^1, D^2 but not $D^1 + D^2$. Suppose we have an initial dual point λ^0 and let $x^0 \in \operatorname{argmin}_x(f + \lambda^0)(x)$ be a solution to the slave subproblem D^1, that is, $D^1(\lambda^0) = f(x^0) + \lambda^0(x^0)$.

Proposition 2.2. Let $\underline{f}$ be a modular minorant of f exact in x^0 and such that $\underline{f} + \lambda^0 \geq D^1(\lambda^0)$ (component-wise). Then the function $\underline{D}^1(\lambda) = \min_x(\underline{f} + \lambda)(x)$ is a minorant of $D^1(\lambda)$ exact at $\lambda = \lambda^0$.

Proof. Since $\underline{f}(x) \leq f(x)$ for all x it follows that $\min_x(\underline{f} + \lambda)(x) \leq \min_x(f + \lambda)(x)$ for all λ and therefore $\underline{D}^1$ is a minorant of D^1. Next, on one hand we have $\underline{D}^1(\lambda^0) \leq D^1(\lambda^0)$ and on the other, $D^1(\lambda^0) \leq (\underline{f} + \lambda^0)(x)$ for all x and thus $D^1(\lambda^0) \leq \underline{D}^1(\lambda^0)$. □

We have constructed a minorant of D^1 which is itself a (simple) piece-wise linear concave function. The maximization step of the minorize-maximize is to solve

$$\max_{\lambda}(\underline{D}^1(\lambda) + D^2(\lambda)). \tag{9}$$

Proposition 2.3. $\lambda^* = -\underline{f}$ is a solution to (9).

Proof. Substituting λ^* into the objective (9) we obtain $\underline{D}^1(\lambda^*) + D^2(\lambda^*) = \min_x(\underline{f} - \underline{f})(x) + D^2(-\underline{f}) = \min_x(\underline{f} + g)(x)$. This value is the maximum because

Algorithm 1: `Primal_MM`

Input: Initial primal point x^k;
Output: New primal point x^{k+2};

1 $\bar{f} \succeq f,\ \bar{f}(x^k) = f(x^k)$; `/* Majorize */`
2 $x^{k+1} \in \operatorname{argmin}_x(\bar{f} + g)(x)$; `/* Minimize */`
3 $\bar{g} \succeq g,\ \bar{g}(x^{k+1}) = g(x^{k+1})$; `/* Majorize */`
4 $x^{k+2} \in \operatorname{argmin}_x(f + \bar{g})(x)$; `/* Minimize */`

[1] $\underline{f}$ reads "f underbar".

Algorithm 2: `Dual_MM`

Input: Initial dual point $\underline{g}^k$;
Output: New dual point $\underline{g}^{k+2}$;

1 $x^k \in \operatorname{argmin}_x(f + \underline{g}^k)(x)$; `/* Minimize */`
`/* Minorize */`
2 $\underline{f}^{k+1} \preceq f$, $\underline{f}^{k+1}(x^k) = f(x^k)$,
$\underline{f}^{k+1} + \underline{g}^k \geq f(x^k) + \underline{g}^k(x^k)$;
3 $x^{k+1} \in \operatorname{argmin}_x(\underline{f}^{k+1} + g)(x)$; `/* Minimize */`
`/* Minorize */`
4 $\underline{g}^{k+2} \preceq g$, $\underline{g}^{k+2}(x^{k+1}) = g(x^{k+1})$,
$\underline{f}^{k+1} + \underline{g}^{k+2} \geq \underline{f}^{k+1}(x^{k+1}) + g(x^{k+1})$;

$\underline{D}^1(\lambda) + D^2(\lambda) = \min_x(\underline{f} + \lambda)(x) + \min_x(g - \lambda)(x) \leq \min_x(\underline{f} + \lambda + g - \lambda)(x) = \min_x(\underline{f} + g)(x)$. □

Note, for the dual point $\lambda = -\underline{f}$, in order to construct a minorant of D^2 (similarly to Proposition 2.2) we need to find a solution to the second slave problem,

$$x^1 \in \operatorname{argmin}(g - \lambda)(x) = \operatorname{argmin}(\underline{f} + g)(x). \quad (10)$$

We obtain Algorithm 2 with the following properties:

- It builds the sequence of dual points given by $\lambda^{2t} = \underline{g}^{2t}, \lambda^{2t+1} = -\underline{f}^{2t+1}$ and the dual objective does not decrease on each step;
- The minimization subproblems and minorants are decoupled (can be solved in parallel) for all horizontal (resp. vertical) chains;
- When provided good minorants (see below) the algorithm has same fixed points as TRW-S [15];
- Updating only a single component λ_i for a pixel i is a monotonous step as well, therefore the algorithm is a *parallel block-coordinate ascent*.

Notice also that `Dual_MM` and `Primal_MM` are very similar, nearly up to replacing minorants with majorants. The sequence $\{E(x^k)\}_k$ is monotonous in Algorithm 1 but not in Algorithm 2.

Good and Fast Minorants The choice of the minorant in `Dual_MM` is non-trivial as there are many, which makes it sort of a secrete ingredient. Figure 3 illustrates two of the possible choices. The *naive* minorant for a chain problem $f + \lambda$ is constructed by calculating its min-marginals and dividing by chain length to ensure that the simultaneous step is monotonous (*c.f.* tree block update algorithm of Sontag and Jaakkola [29, Fig. 1]). The *uniform* minorant is found through the optimization procedure that tries to build the tightest modular lower bound, by increasing uniformly all components that are not yet tight. The details are given in §A. In practice, we build fast minorants, which try to approximate the uniform one using fast message passing operations. Parallelization of decoupled chains allowed us to achieve an implementation which, while having the same number of memory accesses as TRW-S (including messages / dual variables), saturates the GPU memory bandwidth, $\sim$ 230GB/s.[2] This allows to perform 5 iterations of Algorithm 2 for an image 512×512 and 64 labels at the rate of about 30 fps.

[2]This is about 10 times faster than reported for FPGA implementation [7] of TRW-S.

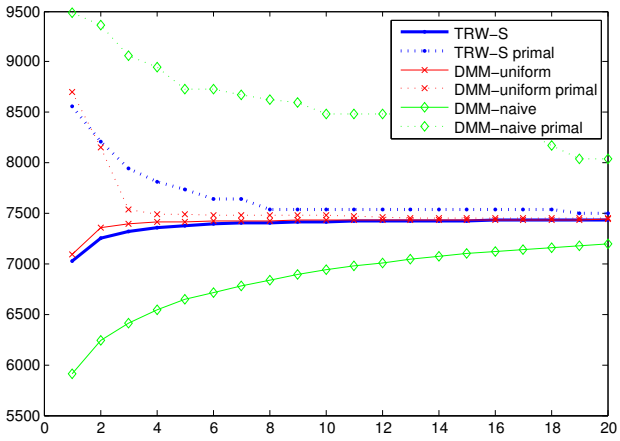


Figure 3: Lower bounds and best primal solutions by TRW-S and by `Dual_MM` with a naive and a uniform minorants. The problem is a small crop from stereo of size 40×40, 16 labels, truncated linear regularization. On the x-axis one iteration is a forward-backward pass of TRW-S *vs.* iteration of `Dual_MM` (equal number of updates per pixel). With a good choice of minorant, `Dual_MM` can perform even better than the sequential baseline in terms of iterations. Parallelizing it can be expected to give a direct speedup.

2.4. Continuous Refinement

In this section we describe the continuous refinement method, which is based on variational energy minimization. The goal of this step is to refine the output of the optimization method described in § 2.3 which is discrete in label-space.

To that end, it is important to minimize the same energy in both formulations. Considering the optimization problem in (1), we are seeking to minimize a non-convex, truncated norm together with a non-convex data term. For clarity, let us write down the problem again:

$$\min_{u \in U} D(u) + R(Au). \quad (11)$$

Non-Convex Primal-Dual Efficient algorithms exist to solve (11) in case both $D(u)$ and $R(Au)$ are convex (but possibly non-smooth), *e.g.* the primal-dual solver of Chambolle and Pock [6]. Kolmogorov et al. [16] solves (11) for a truncated total variation regularizer using a splitting into horizontal and vertical 1D problems and applying [6] to the Lagrangian function. Here we will use a recently proposed extension to [6] by Valkonen [31]. He considers problems of the form $\min_x \mathcal{G}(x) + \mathcal{F}(\mathcal{A}(x))$, *i.e.* of the same structure as (11), where $\mathcal{G}$ and $\mathcal{F}$ are convex, $\mathcal{G}$ is differentiable and $\mathcal{A}(u)$ is a twice differentiable but possibly non-linear operator. In the primal-dual formulation, the problem is written as

$$\min_x \max_y \left[\mathcal{G}(x) + \langle \mathcal{A}(x), y \rangle - \mathcal{F}^*(y)\right], \quad (12)$$

where * is the convex conjugate. Valkonen proposes the following modified primal-dual hybrid gradient method:

$$x^{k+1} = (I + \tau\partial\mathcal{G})^{-1}(x^k - \tau\left[\nabla\mathcal{A}(x^k)\right]^\mathsf{T} y^k) \quad (13a)$$

$$y^{k+1} = (I + \sigma\partial\mathcal{F}^*)^{-1}(y^k + \sigma\mathcal{A}(2x^{k+1} - x^k)). \quad (13b)$$

Reformulation In order to apply method [31], we will reformulate the non-convex problem (11) to the form (12). We start by formulating the regularizer $R(Au)$ as a *difference of convex functions*: $R(Au) = R_+(Au) - R_-(Au)$, where R_+ and R_- are convex. The primal-dual formulation of (11) then reads

$$\min_u \Big[\max_p (\langle Au, p\rangle - R_+^*(p)) + \max_q (\langle Au, q\rangle - R_-^*(q)) + D(u) \Big]. \quad (14)$$

Because $\min_x -f(x) = -\max_x f(x)$, (14) equals

$$\min_u \Big[\max_p (\langle Au, p\rangle - R_+^*(p)) + \min_q (-\langle Au, q\rangle + R_-^*(q)) + D(u) \Big]. \quad (15)$$

Grouping terms we arrive at

$$\min_{u,q} \max_p \Big[\langle Au, p-q\rangle - R_+^*(p) + R_-^*(q) + D(u) \Big]. \quad (16)$$

The problem now arises in minimizing the bilinear term $\langle Au, q\rangle$ in (16) in both u and q. We thus move this term into the nonlinear operator $\mathcal{A}(x)$ and rewrite (16) as

$$\underbrace{\min_{u,q}}_{x} \underbrace{\max_{p,d=1}}_{y} \left\langle \underbrace{\begin{bmatrix} Au \\ -\langle Au, q\rangle \end{bmatrix}}_{\mathcal{A}(x)}, \begin{bmatrix} p \\ d \end{bmatrix} \right\rangle + \underbrace{R_-^*(q) + D(u)}_{\mathcal{G}(x)} - \underbrace{R_+^*(p)}_{\mathcal{F}^*(y)} \quad (17)$$

by introducing a dummy variable $d = 1$.

Implementation Details The gradient of $\mathcal{A}$ needed by iterates (13) is given by

$$\nabla\mathcal{A}(x) = \begin{bmatrix} A & 0 \\ -A^\mathsf{T} q & -Au \end{bmatrix}. \quad (18)$$

The regularization function r is represented as a difference of two convex functions (see Figure 2):

$$r(t) = r_{\varepsilon,\delta}(t) - r_{0,(C+\delta-\varepsilon\delta)}(t), \quad (19)$$

where

$$r_{\alpha,\beta}(t) = \begin{cases} \alpha|t| & \text{if } |t| \le \beta \\ |t| - \beta(1-\alpha) & \text{else} \end{cases} \quad (20)$$

is convex for $\alpha \le 1$. Convex functions $R_+(Au)$ and $R_-(Au)$ are defined by decomposition (19) and (2).

To compute the proximal map $(I + \sigma\partial\mathcal{F}^*)^{-1}(\hat{y})$ we first need the convex conjugate of $\omega_{ij} r_{\alpha,\beta}(t)$. It is given by $(\omega_{ij} r_{\alpha,\beta})^*(t^*) =$

$$\begin{cases} \max(0, \beta|t^*| - \omega_{ij}\alpha\beta) & \text{if } \alpha < |t^*| < \omega_{ij} \\ \infty & \text{else} \end{cases}. \quad (21)$$

The proximal map for $(\omega_{ij} r_{\alpha,\beta})^*$ at $t^* \in \mathbb{R}$ is given by $\bar{t} = \operatorname{clamp}(\pm\omega_{ij}, t')$, where $\operatorname{clamp}(\pm\omega_{ij}, \cdot)$ denotes a clamping to the interval $[-\omega_{ij}, \omega_{ij}]$ and

$$t' = \begin{cases} t^* & \text{if } |t^*| \le \alpha\omega_{ij} \\ \max(\alpha\omega_{ij}, |t^*| - \beta\sigma)\operatorname{sign}(t^*) & \text{else.} \end{cases} \quad (22)$$

Proximal map $(I + \sigma\partial\mathcal{F}^*)^{-1}(\hat{y})$ is calculated by applying expression (22) component-wise to $\hat{y}$. The proximal map $(I + \tau\partial\mathcal{G})^{-1}$ depends on the choice of the data term $D(u)$ and will thus be defined in § 3.

3. Applications

3.1. Stereo Reconstruction

For the problem of estimating depth from two images, we look at a setup of two calibrated and synchronized cameras. We assume that the input images to our method have been rectified according to the calibration parameters of the cameras. We aim to minimize the energy (1) where u encodes the disparity in x-direction. The data term measures the data fidelity between images I_1 and I_2, warped by the disparity field u. As a data term we use the Census Transform [37] computed on a small local patch in each image. The cost is given by the pixel-wise Hamming distance on the transformed images.$D(u)$ is non-convex in the argument u which makes the optimization problem in (1) intractable in general.

We start by minimizing (1) using the discrete method (§2.3) in order to obtain an initial solution $\mathring{u}$. We approximate the data term around the current point $\mathring{u}$ by a piecewise linear convex function $\tilde{D}(u) =$

$$D(\mathring{u}) + \delta_{[\mathring{u}-h, \mathring{u}+h]}(u) + \begin{cases} s_1(u - \mathring{u}) & \text{if } u \le \mathring{u} \\ s_2(u - \mathring{u}) & \text{otherwise} \end{cases} \quad (23)$$

with $s_1 = \frac{D(\mathring{u}+h) - D(\mathring{u})}{h}$ and $s_2 = \frac{D(\mathring{u}) - D(\mathring{u}+h)}{h}$ for a small h. To ensure convexity, we set $s_1 = s_2 = \frac{s_1+s_2}{2}$ if $s_2 < s_1$. The indicator function δ is added to ensure that the solution stays within $\mathring{u} \pm h$ where the approximation is valid. We then apply the continuous method (§2.4). The proximal map $\bar{u} = (I + \tau\partial\mathcal{G})^{-1}(\hat{u})$ needed by the algorithm (13) for the approximated data term expresses as the pointwise soft-thresholding

$$\bar{u}_i = \operatorname{clamp}\left(\mathring{u}_i \pm h, \hat{u}_i - \begin{cases} \tau s_{1,i} & \text{if } \hat{u}_i > \mathring{u}_i + \tau s_{1,i} \\ \tau s_{2,i} & \text{if } \hat{u}_i < \mathring{u}_i + \tau s_{2,i} \\ 0 & \text{otherwise} \end{cases} \right)$$

In practice, the minimization has to be embedded in a *warping framework*: after optimizing for n iterations, the data term is approximated anew at the current solution u.

3.2. Optical Flow

The optical flow problem for two images I_1, I_2 is posed again as model (1). In contrast to stereo estimation, we now have $u_i \in \mathbb{R}^2$ encoding the flow vector. For the discrete optimization step (§2.3) the flow problem is decoupled into two independent stereo-like problems as discussed in §2.2.

For the continuous refinement step, the main problem is again the non-convexity of the data term. Instead of a convex approximation with two linear slopes we build a quadratic approximation, now in 2D, following [34]. The approximated data term reads $\tilde{D}_i(u_i) = \delta_{[\mathring{u}_i - h, \mathring{u}_i + h]}(u_i) +$

$$D_i(\mathring{u}_i) + L_i^\mathsf{T}(u_i - \mathring{u}_i) + \frac{1}{2}(u_i - \mathring{u}_i)^\mathsf{T} Q_i (u_i - \mathring{u}_i), \quad (24)$$

where $L_i \in \mathbb{R}^2$ and $Q_i \in \mathbb{R}^{2\times 2}$ are finite difference approximations of the gradient and the Hessian with step-size h. Convexity of (24) is ensured by retaining only positive-semidefinite part of Q_i as in [34]. The proximal map $\bar{u} = (I + \tau \partial \mathcal{G})^{-1}(\hat{u})$ for data term (24) is given point-wise by

$$\bar{u}_i^k = \text{clamp}\left(\mathring{u}_i^k \pm h, \frac{\hat{u}_i^k + \tau(Q_i \mathring{u}_i - L_i)^k}{1 + \tau L_i^k}\right). \quad (25)$$

Optimizing (1) is then performed as proposed in §2.4.

4. Experiments

4.1. Stereo Reconstruction

We evaluate our proposed real-time stereo method on datasets where Ground-Truth data is available as well as on images captured using a commercially available stereo camera.

4.1.1 Influence of Truncated Regularizer

We begin by comparing the proposed method to a simplified version that does not use a truncated norm as regularizer but a standard Total Variation. We show the effect of this change in Fig. 4, where one can observe much sharper edges, when using a robust norm in the regularization term. On the downside it is more sensitive to outliers, which however can be removed in a post-processing step like a two-side consistency check.

4.1.2 Live Dense Reconstruction

To show the performance of our stereo matching method in a real live setting, we look at the task of creating a live dense reconstruction from a set of depth images. To that end, we are using a reimplementation of *KinectFusion* proposed by Newcombe et al. [20] together with the output of our method. This method was originally designed to be used with the RGBD output of a Microsoft Kinect and tracks the 6 DOF position of the camera in real-time.

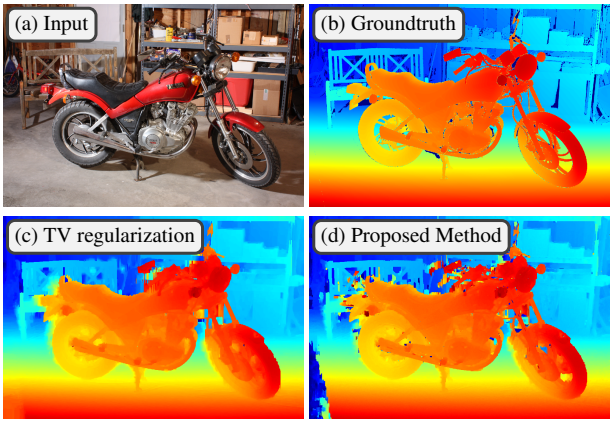


Figure 4: Influence of the robust regularizer in the continuous refinement on stereo reconstruction quality.

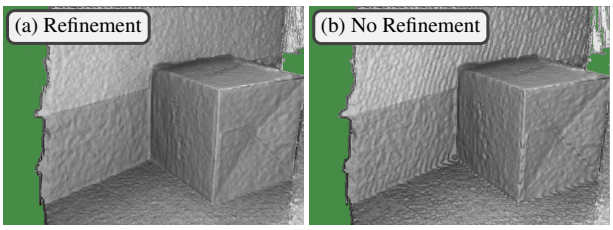


Figure 5: Influence of continuous refinement on the reconstruction quality of KinectFusion.

For the purpose of this experiment we replace the Kinect with a Point Grey Bumblebee2 stereo camera. *KinectFusion* can only handle relatively small camera movements between images, so a high framerate is essential. We set the parameters to our method to achieve a compromise between highest quality and a framerate of $\approx 4-5$ fps: camera resolution 640×480, 128 disparities, 4 iterations of `Dual_MM`, 5 warps and 40 iterations per warp of the continuous refinement.

Influence of Continuous Refinement The first stage of our reconstruction method, `Dual_MM`, already delivers high quality disparity images that include details on fine structures and depth discontinuities that are nicely aligned with edges in the image. In this experiment we want to show the influence of the second stage, the continuous refinement, on the reconstruction quality of KinectFusion. To that end we mount the camera on a tripod and collect 300 depthmaps live from our full method and 300 frames with the continuous refinement switched off. By switching off the camera tracking, the final reconstruction will show us the artifacts produced by the stereo method. Figure 5 depicts the result of this comparison. One can easily see that the output of the discrete method contains fine details, but suffers from staircasing artifacts on slanted surfaces due to the integer solution. The increase in quality due to the refinement stage can be especially seen on far away objects, where a disparity step of 1 pixel is not enough to capture smooth surfaces.

Timing To show the influence of the individual steps in our stereo method on runtime, we break down the total

Cost Vol.	Discrete	Cont. Ref.	Total
27 ms	73 ms	39 ms	**139 ms**

Table 1: Runtime analysis of the individual components of our stereo matching method. Details regarding computing hardware and parameters are in the text. In case of the full left-right check procedure the total computation time doubles.

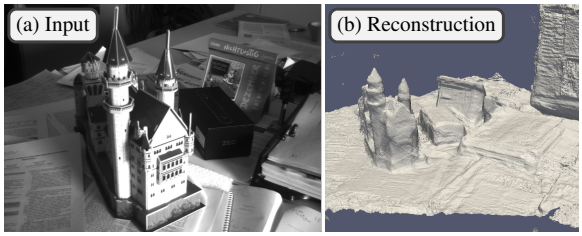


Figure 6: Qualitative result of reconstructing a desktop scene using KinectFusion[3].

time of $\approx$ 140 ms per frame in Table 1. Those timings have been achieved using a PC with 32 GB RAM with a NVidia 980GTX, running Linux.

Qualitative Results To give an impression about the quality of the generated depthmaps and the speed of our method, we run our full algorithm and aim to reconstruct a desktop scene with a size of $1 \times 1 \times 1$ meters and show some renderings in Fig. 6. To better visualize the quality of the geometry, the model is rendered without texture[3].

4.2. Optical Flow

In this section we show preliminary results of our algorithm applied to optical flow. A further improvement in quality can be expected by exploiting the coupled scheme [26] in the discrete optimization, as discussed in § 2.2. As depicted in Figure 7, our method is able to deliver reasonable results on a variety of input images. We deliberately chose scenes that contain large motion as well as small scale objects, to highlight the strengths of the discrete-continuous approach. For comparison, we use a state-of-the-art purely continuous variational optical flow algorithm [33]. The runtime of our method is 2s for an image of size 640×480.

5. Conclusion

The current results demonstrate that it is feasible to solve dense image matching problems using global optimization methods with a good quality in real time. We have proposed a highly parallel discrete method, which even when executed sequentially, is competitive with the best sequential methods. As a dual method, we believe, it has a potential to smoothly handle more complex models in the dual decomposition framework and is in theory applicable to general graphical models. When the solution is sufficiently localized, continuous representation increases the accuracy of the model as well as optimization speed. In the continuous optimization, we experimented with non-convex models and showed a reduction allowing to handle them with the help of a recent non-linear primal-dual method. This in turn allowed to speak of a global model to be solved by a discrete-continuous optimization.

Ideally, we would like to achieve a method, which, when given enough time, produces an accurate solution, and in the real time setting gives a robust result. We plan further to improve on the model. A vast literature on the topic suggest that modeling occlusions and using planar hypothesis can be very helpful. At the same time, we are interested in a tighter coupling of discrete and continuous optimization towards a globally optimal solution.

[3]We point the interested reader to a video that shows the reconstruction pipeline in real-time: `http://gpu4vision.icg.tugraz.at/videos/cvww16.mp4`

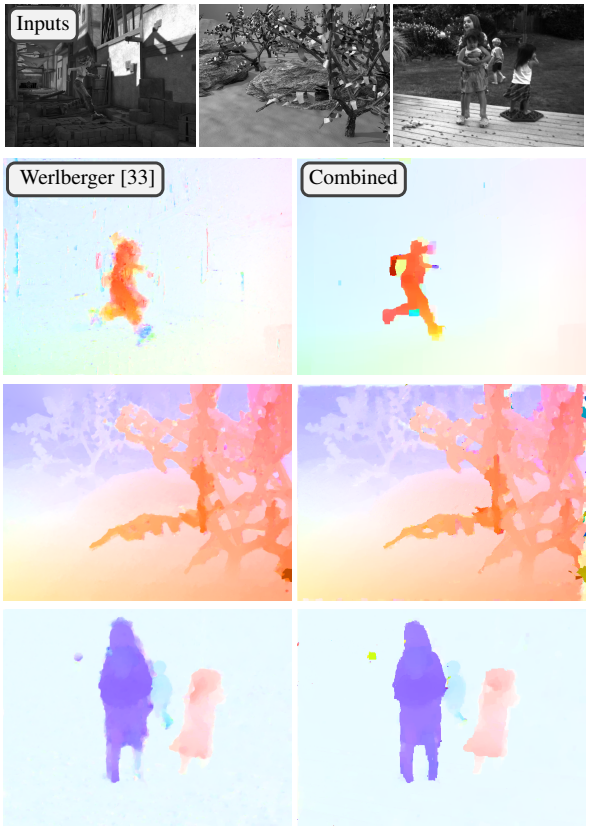


Figure 7: Subjective comparison of variational approach [33] (left) with our combined method (right). Top row show input images, one from a pair. Both methods use the same data term. Parameters of both algorithms have been tuned by hand to deliver good results. Note that for [33] it is often impossible to get sharp motion boundaries as well as small scale details, despite a very strong data term (*e.g.* artifacts in left image, first row).

Acknowledgements

This work was supported by the research initiative Mobile Vision with funding from the AIT and the Austrian Federal Ministry of Science, Research and Economy HRSM programme (BGBl. II Nr. 292/2012).

References

[1] Arashloo, S. R. and Kittler, J. (2014). Fast pose invariant face recognition using super coupled multiresolution Markov random fields on a GPU. *Pattern Recognition Letters*, 48.

[2] Banz, C., Hesselbarth, S., Flatt, H., Blume, H., and Pirsch, P. (2010). Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. In *ICSAMOS*.

[3] Bleyer, M. and Gelautz, M. (2008). Simple but effective tree structures for dynamic programming-based stereo matching. In *VISAPP*.

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

[5] Brox, T., Bruhn, A., and Weickert, J. (2006). Variational motion segmentation with level sets. In *ECCV*, volume 3951.

[6] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1).

[7] Choi, J. and Rutenbar, R. A. (2012). Hardware implementation of MRF MAP inference on an FPGA platform. In *Field Programmable Logic*.

[8] Combettes, P. L. and Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*.

[9] Drory, A., Haubold, C., Avidan, S., and Hamprecht, F. (2014). Semi-global matching: A principled derivation in terms of message passing. In *Pattern Recognition*, volume 8753.

[10] Facciolo, G., de Franchis, C., and Meinhardt, E. (2015). MGM: A significantly more global matching for stereovision. In *BMVC*.

[11] Hirschmuller, H. (2011). Semi-global matching-motivation, developments and applications.

[12] Hosni, A., Rhemann, C., Bleyer, M., Rother, C., and Gelautz, M. (2013). Fast cost-volume filtering for visual correspondence and beyond. *PAMI*, 35(2).

[13] Hurkat, S., Choi, J., Nurvitadhi, E., Martınez, J. F., and Rutenbar, R. A. (2012). Fast hierarchical implementation of sequential tree-reweighted belief propagation for probabilistic inference. In *Field Programmable Logic*.

[14] Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Lellmann, J., Komodakis, N., and Rother, C. (2013). A comparative study of modern inference techniques for discrete energy minimization problem. In *CVPR*.

[15] Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28(10).

[16] Kolmogorov, V., Pock, T., and Rolinek, M. (2015). Total variation on a tree. *CoRR*, abs/1502.07770.

[17] Lawler, E. (1966). Optimal cycles in doubly weighted directed linear graphs. In *Intl Symp. Theory of Graphs*.

[18] Menze, M., Heipke, C., and Geiger, A. (2015). Discrete optimization for optical flow. In *GCPR*.

[19] Narasimhan, M. and Bilmes, J. (2005). A supermodular-submodular procedure with applications to discriminative structure learning. In *Uncertainty in Artificial Intelligence*.

[20] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*.

[21] Ochs, P., Chen, Y., Brox, T., and Pock, T. (2014). ipiano: Inertial proximal algorithm for non-convex optimization. *SIAM JIS*, 7(2).

[22] Prusa, D. and Werner, T. (2015). Universality of the local marginal polytope. *PAMI*, 37(4).

[23] Ranftl, R., Bredies, K., and Pock, T. (2014). Non-local total generalized variation for optical flow estimation. In *ECCV*.

[24] Ranftl, R., Gehrig, S., Pock, T., and Bischof, H. (2012). Pushing the limits of stereo using variational stereo estimation. In *Intelligent Vehicles Symposium*.

[25] Roth, S., Lempitsky, V., and Rother, C. (2009). Discrete-continuous optimization for optical flow estimation. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, volume 5604.

[26] Shekhovtsov, A., Kovtun, I., and Hlaváč, V. (2008). Efficient MRF deformation model for non-rigid image matching. *CVIU*, 112.

[27] Shekhovtsov, A., Swoboda, P., and Savchynskyy, B. (2015). Maximum persistency via iterative relaxed inference with graphical models. In *CVPR*.

[28] Sinha, S. N., Scharstein, D., and Szeliski, R. (2014). Efficient high-resolution stereo matching using local plane sweeps. In *CVPR*.

[29] Sontag, D. and Jaakkola, T. S. (2009). Tree block coordinate descent for MAP in graphical models. In *AISTATS*.

[30] Taniai, T., Matsushita, Y., and Naemura, T. (2014). Graph cut based continuous stereo matching using locally shared labels. In *CVPR*.

[31] Valkonen, T. (2014). A primal-dual hybrid gradient method for nonlinear operators with applications to MRI. *Inverse Problems*, 30(5).

[32] Wainwright, M., Jaakkola, T., and Willsky, A. (2005). MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11).

[33] Werlberger, M. (2012). *Convex Approaches for High Performance Video Processing*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria.

[34] Werlberger, M., Pock, T., and Bischof, H. (2010). Motion estimation with non-local total variation regularization. In *CVPR*.

[35] Werner, T. (2007). A linear programming approach to max-sum problem: A review. *PAMI*, 29(7).

[36] Woodford, O., Torr, P., Reid, I., and Fitzgibbon, A. (2009). Global stereo reconstruction under second-order smoothness priors. *PAMI*, 31(12).

[37] Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *ECCV*, volume 801.

Appendix A. Details of Dual MM

In this section we specify details regarding computation of minorants in `Dual_MM`. The minorants are computed using message passing and we'll also need the notion of min-marginals.

A.1. Min-Marginals and Message Passing

Definition A.1. For cost vector f its *min-marginal* at node i is the function $m^f : \mathcal{L} \to \mathbb{R}$ given by

$$m^f(x_i) = \min_{x_{\mathcal{V}\setminus i}} f(x). \tag{26}$$

Function $m^f(x_i)$ is a min projection of $f(x)$ onto x_i only. Given the choice of x_i, it returns the cost of the best labeling in f that passes through x_i. For a chain problem it can be computed using dynamic programming. Let us assume that the nodes $\mathcal{V}$ are enumerated in the order of the chain and $\mathcal{E} = \{(i, i+1) \,|\, i = 1 \dots |\mathcal{V}| - 1\}$. We then need to compute: *left min-marginals*: $\varphi_{i-1,i}(x_i) :=$

$$\min_{x_{1,\dots i-1}} \sum_{i'<i} f_{i'}(x_{i'}) + \sum_{i'j' \in \mathcal{E} \,|\, i'<i} f_{i'j'}(x_{i'}, x_{j'}); \tag{27}$$

and *right min-marginals*: $\varphi_{i+1,i}(x_i) :=$

$$\min_{x_{i+1,\dots |V|}} \sum_{i'>i} f_{i'}(x_{i'}) + \sum_{i'j' \in \mathcal{E} \,|\, i' \geq i} f_{i'j'}(x_{i'}, x_{j'}). \tag{28}$$

These values for all $ij \in \mathcal{E}$, $x_i, x_j \in \mathcal{L}$ can be computed dynamically (recursively). After that, the min-marginal $m^f(x_i)$ expresses as

$$m^f(x_i) = f_i(x_i) + \varphi_{i-1,i}(x_i) + \varphi_{i+1,i}(x_i). \tag{29}$$

TRW-S method [15] can be derived as selecting one node i at a time and maximizing (8) with respect to λ_i only. For the two slave problems in (8) TRW-S needs to compute min-marginals $m^{f+\lambda}(x_i)$ and $m^{g-\lambda}(x_i)$. A (non-unique) optimal choice for λ_i would be to ensure that

$$m^{f+\lambda}(x_i) = m^{g-\lambda}(x_i) \;\; \forall x_i \in \mathcal{L} \tag{30}$$

by setting

$$\lambda_i := \lambda_i + (m^{g-\lambda}(x_i) - m^{f+\lambda}(x_i))/2. \tag{31}$$

If i and j are two nodes in a chain $f+\lambda$ then performing the update of λ_i changes the min-marginal at j and vice-versa. The updates must be implemented sequentially or otherwise one gets a non-monotonous behavior and the method may fail to converge (see [15]).

TRW-S gains its efficiency in that after the update (31), the min-marginal at a neighboring node can be recomputed by a single step of dynamic programming. Let the neighboring node be $j = i + 1$. The expression for the right min-marginal at j remains correct and the expression for left min-marginal is updated using its recurrent expression $\varphi_{ij}(x_j) :=$

$$\min_{x_i} \big[\varphi_{i-1,i}(x_i) + f_i(x_i) + f_{ij}(x_i, x_j)\big], \tag{32}$$

also known as *message passing*. Then min-marginal at j becomes available through (29).

It is possible to perform update (31) in parallel by scaling down the step size by the number of variables (or the length of the chain). This is equivalent to decomposing a chain f into n copies with costs f/n so that they contribute one for each node i with a min-marginal $m^f(x_i)/n$. Confer to the parallel tree block update algorithm of Sontag and Jaakkola [29, Fig. 1]). However, the gain from the palatalization does not pay off the decrease in the step size.

A.2. Slacks

In the following we will also use the term slack. Shortly, it is explained as follows. The dual problem (8) can be written as a linear program, see *e.g.*, [35]. Dual inequality constraints in that program can satisfied as equalities, in which case they are tight, or they can be satisfied as strict inequalities in which case there is a *slack*. Equivalent reparametrization of the problem (change of the dual variables) can propagate a slack from one constraint (corresponding to a label-node pair) to another one. If all constraints in a group becomes non-tight, their minimum slack can be subtracted and increments the lower bound. Since for a chain problem the LP relaxation is tight, the maximum slack that can be concentrated in a label-node equals the corresponding min-marginal.

A.3. Good Minoratns

Definition A.2. A modular minorant λ of f is *maximal* if there is no other modular minorant $\lambda' \geq \lambda$ such that $\lambda'(x) > \lambda(x)$ for some x.

Lemma A.3. For a maximal minorant λ of f all min-marginals of $f - \lambda$ are identically zero.

Proof. Since λ is a minorant, min-marginals $m_i(x_i) = \min_{x_{\mathcal{V}\setminus i}}[f(x) - \lambda(x)]$ are non-negative. Assume for contradiction that $\exists i$, $\exists x_i$ such that $m_i(x_i) > 0$. Clearly, $\lambda'(x) := \lambda(x) + m_i(x_i)$ is also a minorant and $\lambda' > \lambda$. □

Even using maximal minorants, the Algorithm 2 can get stuck in fixed points which do not satisfy weak tree

agreement [15], *e.g.* suboptimal even in the class of message passing algorithms. Consider the following example of a minorant leading to a poor fixed point.

Example A.4. Consider a model in Figure 8 with two labels and strong Ising interactions ensuring that the optimal labeling is uniform. If we select minorants that just takes the unary term, without redistributing it along horizontal or vertical chains, the lower bound will not increase. For example, for the horizontal chain (v_1, v_2), the minorant $(1,\ 0)$ (displayed values correspond to $\lambda_v(1) - \lambda_v(2)$). This minorant is maximal, but it does not propagate the information available in v_1 to v_2 for the exchange with the vertical chain (v_2, v_4).

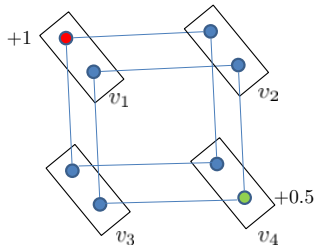


Figure 8: Example minorize-minimize stucks with a minorant that does not redistribute slack.

A.3.1 Uniform Minorants

Dual algorithms, by dividing the slacks between subproblems ensure that there is always a non-zero fraction of it (depending on the choice of weights in the scheme) propagated along each chain. We need a minorant, which will expose in every variable what is the preferable solution for the subproblem. We can even try to treat all variables uniformly. The practical strategy proposed below is motivated by the following.

Proposition A.5. Let $f^* = \min_x f(x)$ and let O_u be the support set of all optimal solutions x^*_u in $u \in \mathcal{V}$. Consider the minorant λ given by $\lambda_u(x_u) = \varepsilon(1 - O_u)$ and maximizing ε:

$$\max\{\varepsilon \mid (\forall x)\ \varepsilon\langle 1 - O, x\rangle \leq f(x)\}. \tag{33}$$

The above minorant assigns cost ε to all labels but those in the set of optimal solutions. If the optimal solution x^* is unique, it takes the form $\lambda = \varepsilon(1 - x^*)$. This minorant corresponds to the direction of the subgradient method and ε determines the step size which ensures monotonicity. However it is not maximal. In $f - \lambda$ there still remains a lot of slack that can be useful when exchanging to the other problem. It is possible to consider $f - \lambda$ again. If we have solved (33), it will necessarily have a larger set of optimal solutions. We can search for a maximal ε_1 that can be subtracted from all non-optimal label-nodes in $f - \lambda$ and so on. The algorithm is specified as Algorithm 3.

Algorithm 3: Maximal Uniform Minorant

Input: Chain subproblem f;
Output: Minorant λ;

```
1 λ := 0;
2 while true
3     Compute min-marginals m of f − λ;
4     if m = 0 then return λ;
5     Let O := ⟦m = 0⟧, the support set of optimal
      solutions of m − λ;
6     Find max{ε | (∀x) ε⟨1 − O, x⟩ ≤ (f − λ)(x)};
7     Let λ := λ + ε(1 − O);
```

The optimization problem in Line 6 can be solved using the minimum ratio cycle algorithm of Lawler [17]. We search for a path with a minimum ratio of the cost given by $(f - \lambda)(x)$ to the number of selected labels with non-zero min-marginals given by $\langle 1 - O, x\rangle$. This algorithm is rather efficient, however Algorithm 3 it is still too costly and not well-suited for a parallel implementation. We will not use this method in practice directly, rather it establishes a sound baseline that can be compared to.

The resulting minorant λ is maximal and uniform in the following sense.

Lemma A.6. Let m be the vector of min-marginals of f. The uniform minorant λ found by Algorithm 3 satisfies

$$\lambda \geq m/n, \tag{34}$$

where n is the length of the longest chain in f.

Proof. This is ensured by Algorithm 3 as in each step the increment ε results from dividing the min-marginal by $\langle 1 - O, x\rangle$ which is at most the length of the chain. □

In fact, when the chain is strongly correlated, the minorant will approach m/n and we cannot do better than that. However, if the correlation is not as strong the minorant becomes tighter, and in the limit of zero pairwise interactions there holds $\lambda = m$. In a sense the minorant computes "decorrellated" min-marginals.

The next example illustrates uniform minorants and steps of the algorithm.

Example A.7. Consider a chain model with the following data unary cost entries (3 labels, 6 nodes):

0 0 1 0 0 8
9 7 0 3 2 8
7 3 6 9 1 0

The regularization is a Potts model with cost $f_{uv}(x_u, x_v) = 1[\![x_u \neq x_v]\!]$. Min-marginals of the problem and iteration of Algorithm 3 are ilustrated in Figure 9. At the first iteration the constructed minorant is

0 0 0 0 0 1
1 1 1 1 1 1
1 1 1 1 1 0

And the final minorant is:

0	0	0	0	0	7
8	7	1	2	2	7
6	4	6	7	1	0

The minorant follows min-marginals (first plot in Figure 9), because the interaction strength is relatively weak and min-marginals are nearly independent. If we increase interaction strength to 5, we find the following min-marginals and minorant, respectively:

0	0	0	0	0	3
14	15	8	8	7	8
12	13	15	10	1	0

0	0	0	0	0	3
5.5	5.5	3	3	3	3
4.75	4.75	4.75	4.75	1	0

It is seen that in this case min-marginals are correlated and only a fraction can be drained in parallel. The uniform approach automatically divides the cost equally between strongly correlated labels.

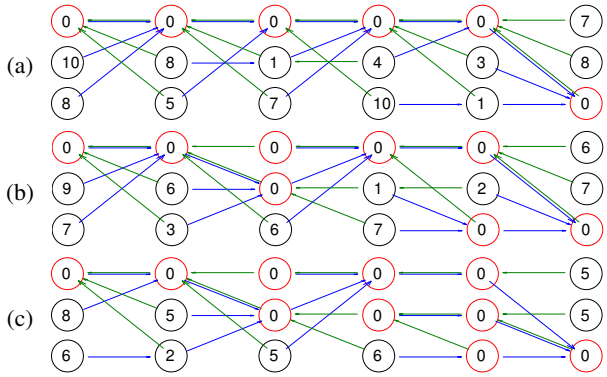


Figure 9: (a) Min-marginals (normalized by subtracting the value of the minimum) at vertices and arrows allowing to backtrack the optimal solution passing through a given vertex. (b), (c) min-marginals of $f - \lambda$ after one (resp. two) iterations of Algorithm 3 ($\varepsilon_1 = 1$ and $\varepsilon_2 = 1$). With each iteration the number of vertices having zero min-marginal strictly increases.

A basic performance test of `Dual_MM` with uniform minorants versus TRW-S is shown in Figure 3. It demonstrates that the `Dual_MM` can be faster, when provided good minorants. The only problem is that determining the uniform minorant involves repeatedly solving minimum ratio path problems, plus there is a numerical instability in determining the support set of optimal solutions O.

A.3.2 Iterative Minorants

A simpler way to construct a maximal minorant would be to iteratively subtract from f a portion of its min-marginals and accumulate them in the minorant, until all min-marginals of the reminder become zero. Algorithm 4 implements this idea. The portion of min-marginals drained from the reminder $f - \lambda$ to the minorant λ in each iteration is controlled by $\gamma_s \in (0, 1]$. Reversing the chain

Algorithm 4: Iterative Minorant

Input: Chain subproblem f;
Output: Minorant λ;

```
λ := 0;
for s = 1 ... max_pass do
    for i = 1 ... |V| do
        Compute min-marginal m_i of f − λ at i
        dynamically, equations (32) and (29);
        λ_i += γ_s m_i;
    Reverse the chain;
```

efficiently alternates between the forward and the backward passes. For the last pass coefficient γ_s is set to 1 to ensure that the output minorant is maximal. Figure 10 illustrates that this idea can perform well in practice.

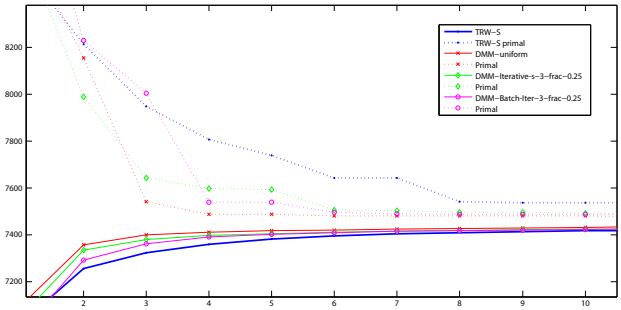


Figure 10: Same setting as in Figure 3. The new plots show that Iterative minorants are not as good as uniform but still perform very well. Parameter `max_pass` $= 3$ and $\gamma_s = 0.25$ were used. The Batch Iterative method (Batch-Iter) runs forward-backward iterations in a smaller range, which is more cache-efficient and is also performing relatively well in this example.

A.3.3 Hierarchical Minorants

The idea of hierarchical minorants is as follows. Let f be a one horizontal chain. We can break it into two subchains of approximately the same size, sharing a variable x_i in the middle. By introducing a Lagrange multiplier over this variable, we can decouple the two chains. The value of the Lagrange multiplier can be chosen such that both subchains have exactly the same min-marginals in x_i. This makes the split uniform in a certain sense. Proceeding so we increase the amount of parallelism and hierarchically break the chain down to two-variable pieces, for which the minorant is computed more or less straightforwardly. This is the method used to obtain all visual experiments in the paper. Its more detailed benchmarking is left for future work. We detail now the simplest case when the chain has length two, *i.e.*, the energy is given by $f_1(x_1) + f_{12}(x_1, x_2) + f_2(x_2)$. The procedure to compute the minorant is as follows:

- Compute $m_1^f(x_1)$ and let $\lambda_1 := m_1^f(x_1)/2$. *I.e.*, we subtract a half of the min-marginal in the first node.
- Recompute the new min-marginal at node 2: update

Algorithm 5: `Handshake`

Input: Energy terms f_i, f_j, f_{ij}, messages $\varphi_{i-1,i}(x_i)$ and $\varphi_{j,j+1}(x_j)$;
Output: Messages for decorrellated chains: $\varphi_{ji}(x_i)$ and $\varphi_{ij}(x_j)$;

`/* Message from j to i */`
$\varphi_{ji}(x_i) := \texttt{Msg}_{ji}(f_j + \varphi_{j,j+1})$;
`/* Total min-marginal at i */`
$m_i(x_i) := \varphi_{i-1,i}(x_i) + f_i(x_i) + \varphi_{ji}(x_i)$;
`/* Share a half to the right */`
$\varphi_{ij}(x_j) := \texttt{Msg}_{ij}(m_i/2 - \varphi_{ji})$;
`/* Bounce back what cannot be shared */`
$\varphi_{ji}(x_i) := \texttt{Msg}_{ji}(-\varphi_{ij})$;
Procedure $\texttt{Msg}_{ij}(a)$
  Input: Unary cost $a \in \mathbb{R}^K$;
  Output: Message from i to j;
  return $\varphi(x_j) := \min_{x_i \in \mathcal{L}} \left[a(x_i) + f_{ij}(x_i, x_j)\right]$;

the message $\varphi_{12}(x_2) := \texttt{Msg}_{12}(f_1 - \lambda_1)$; Reassemble $m_2^{f-\lambda}(x_2) = \varphi_{12}(x_2) + f_2(x_2)$.

- Take this whole remaining min-marginal to the minorant: let $\lambda_2 := m_2^{f-\lambda}(x_2)$.
- Recompute the new min-marginal at node 1: update the message $\varphi_{21}(x_1) := \texttt{Msg}_{21}(f_2 - \lambda_2)$; It still may be non-zero. For example, if the pairwise term of f is zero we recover the remaining half of the initial min-marginal at node 1. Let $\lambda_1 \mathrel{+}= m_1^{f-\lambda}(x_1)$.

Importantly, the computation has been expressed in terms of message passing, and therefore can be implemented as efficiently. The procedure fro the two-node case is straightforwardly generalized to longer chains. Let ij be an edge in the middle of the chain. We compute left min-marginal at i, right min-marginal at j and then apply the `Handshake` procedure over the edge ij, defined in Algorithm 5. The procedure divides the slack between nodes i and j similarly to how it is described above for the pair. The result of this redistribution is encoded directly in the messages. The two subchains $1, \dots i$ and $j, \dots |\mathcal{V}|$ are "decorrellated" by the `Handshake` and will not talk to each other further during the construction of the minorant. The left min-marginal for subchain $j, \dots |\mathcal{V}|$ at node $j+1$ is computed using update (32) and so on until the middle of the subchain where a new `Handshake` is invoked. The minorant is computed at the lowest level of hierarchy when the length of the subchain becomes two. The structure of the processing is illustrated in Figure 11. It is seen that each level after the top one requires to send messages only for a half of nodes in total. Moreover, there is only a logarithmic number of level. It turns out that this procedure is not much more computationally costly than just computing min-marginals. For example, to restore left min-marginal for the subchain $j, \dots |V|$, in node $i+1$ we

We conjecture that while iterative minorants may transfer only a geometric fraction of min-marginals in some cases, the hierarchical minorant is only by a constant factor inferior to the uniform one.

```
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>]
[.......<<<<<<<<][>>>>>>>>.......]
[...<<<][>>>...][...<<<][>>>...]
[.<][>.][.<][>.][.<][>.][.<][>.]
[][][][][][][][][][][][][][][][]
```

Figure 11: Messages passed in the construction of the hierarchical minorant for a chain of length 32. From top to bottom: level of hierarchical processing. Symbols > and < denote message passing in the respective direction. Brackets `[]` mark the limits of the decorrellated sub-chains at the current level. Dots denote places where the previously computed messages in the needed direction remain valid and need not be recomputed. Places where the two opposite messages meet correspond to the execution of the `Handshake` procedure. The lowest level consists of 16 decorrellated chains of length 2 each.

A.4. Iteration Complexity

The bottleneck in a fast implementation of dual algorithms are the memory access operations. This is simply because there is a big cost data volume that needs to be scanned in each iteration plus messages have to be red and written in TRW-S as well as in out Algorithm 2 (dual variables λ). We therefore will assess complexity in terms of memory access operations and ignore the slightly higher arithmetic complexity of our minorants.

For TRW-S the accesses per pixel are:

- read all incoming messages (4 access);
- read data term (1 access);
- write out messages in the pass direction (2 accesses).

The cache can potentially amortize writing messages and reading them back in the next scan line, in which case the complexity could be counted as 5 accesses per pixel. However, currently only CPU cache is big enough for this, while multiprocessors in GPU have relatively small cache divided between many parallel threads.

For the iterative minorant we have 3 forward-backward passes reading the data cost, the reverse message and writing the forward message (3*2*3 accesses), the last iteration writes λ and not the message. Some saving is possible with a small cache set at a cost of more computations. Computing the hierarchical minorant as described in Figure 11 for a chain of length 2048, assuming that chunks of size 8 already fit in the fast memory (registers + shared memory) has the following complexity. Reading data costs and writing messages until length 8 totals to $2 + \log_2(2048/8)/2 = 6$ accesses. Reading messages is only required at `Handshake` points and needs to be counted only until reaching length 8. Writing λ adds one more access. These estimates are summarized in Table 2.

TRW-S	Iterative	Naive BCD	Hierarchical
7(5)	18(8)	5(4)	7

Table 2: Memory accesses per pixel in TRW-S and `Dual_MM` with variants of minorants. Naive BCD here means just computing min-marginals.