Fast and Differentiable Message Passing on Pairwise Markov Random Fields

Zhiwei Xu[1,2][0000−0001−8283−6095], Thalaiyasingam Ajanthan[1][0000−0002−6431−0775], and Richard Hartley[1][0000−0002−5005−0191]

[1] Australian National University and Australian Centre for Robotic Vision
[2] Data61, CSIRO, Canberra, Australia
{firstname.lastname}@anu.edu.au

Abstract. Despite the availability of many Markov Random Field (MRF) optimization algorithms, their widespread usage is currently limited due to imperfect MRF modelling arising from hand-crafted model parameters and the selection of inferior inference algorithm. In addition to differentiability, the two main aspects that enable learning these model parameters are the forward and backward propagation time of the MRF optimization algorithm and its inference capabilities. In this work, we introduce two fast and differentiable message passing algorithms, namely, Iterative Semi-Global Matching Revised (ISGMR) and Parallel Tree-Reweighted Message Passing (TRWP) which are greatly sped up on a GPU by exploiting massive parallelism. Specifically, ISGMR is an iterative and revised version of the standard SGM for general pairwise MRFs with improved optimization effectiveness, and TRWP is a highly parallel version of Sequential TRW (TRWS) for faster optimization. Our experiments on the standard stereo and denoising benchmarks demonstrated that ISGMR and TRWP achieve much lower energies than SGM and Mean-Field (MF), and TRWP is two orders of magnitude faster than TRWS without losing effectiveness in optimization. We further demonstrated the effectiveness of our algorithms on end-to-end learning for semantic segmentation. Notably, our CUDA implementations are at least 7 and 700 times faster than PyTorch GPU implementations for forward and backward propagation respectively, enabling efficient end-to-end learning with message passing.

1 Introduction

Optimization of Markov Random Fields (MRFs) has been a well-studied problem for decades with a significant impact on many computer vision applications such as stereo vision [1], image segmentation [2], texture modeling [3]. The widespread use of these MRF optimization algorithms is currently limited due to imperfect MRF modelling [4] because of hand-crafted model parameters, the usage of inferior inference methods, and non-differentiability for parameter learning. Thus, better inference capability and computing efficiency are essential to improve its performance on optimization and modelling, such as energy optimization and end-to-end learning.

Even though parameter and structural learning with MRFs has been employed successfully in certain cases, well-known algorithms such as Mean-Field (MF) [5,6] and Semi-Glocal Matching (SGM) [7], are suboptimal in terms of optimization capability.

Specifically, the choice of an MRF algorithm for optimization is driven by its inference ability, and for learning capability through efficient forward and backward propagation and parallelization capabilities.

In this work, we consider message passing algorithms due to their generality, high inference ability, and differentiability, and provide efficient CUDA implementations of their forward and backward propagation by exploiting massive parallelism. In particular, we revise the popular SGM method [1] and derive an iterative version noting its relation to traditional message passing algorithms [8]. In addition, we introduce a highly parallelizable version of the state-of-the-art Sequential Tree-Reweighted Message Passing (TRWS) algorithm [9], which is more efficient than TRWS and has similar minimum energies. For both these methods, we derive efficient backpropagation by unrolling their message updates and cost aggregation and discuss massively parallel CUDA implementations which enable their feasibility in end-to-end learning.

Our experiments on the standard stereo and denoising benchmarks demonstrate that our Iterative and Revised SGM method (ISGMR) obtains much lower energies compared to the standard SGM and our Parallel TRW method (TRWP) is two orders of magnitude faster than TRWS with virtually the same minimum energies and that both outperform the popular MF and SGM inferences. Their performance is further evaluated by end-to-end learning for semantic segmentation on PASCAL VOC 2012 dataset.

Furthermore, we empirically evaluate various implementations of the forward and backward propagation of these algorithms and demonstrate that our CUDA implementation is the fastest, with *at least 700 times speed-up* in backpropagation compared to a PyTorch GPU version. Code is available at *https://github.com/zwxu064/MPLayers.git*.

Contributions of this paper can be summarised as:

- We introduce two message passing algorithms, ISGMR and TRWP, where ISGMR has higher optimization effectiveness than SGM and TRWP is much faster than TRWS. Both of them outperform the popular SGM and MF inferences.
- Our ISGMR and TRWP are massively parallelized on GPU and can support any pairwise potentials. The CUDA implementation of the backpropagation is at least 700 times faster than the PyTorch auto-gradient version on GPU.
- The differentiability of ISGMR and TRWP is presented with gradient derivations, with effectiveness validated by end-to-end learning for semantic segmentation.

2 Related Work

In MRF optimization, estimating the optimal latent variables can be regarded as minimizing a particular energy function with given model parameters. Even if the minimum energy is obtained, high accuracy cannot be guaranteed since the model parameters of these MRFs are usually handcrafted and imperfect. To tackle this problem, learning-based methods were proposed. However, most of these methods rely greatly on finetuning the network architecture or adding learnable parameters to increase the fitting ability with ground truth. This may not be effective and usually requires high GPU memory.

Nevertheless, considering the highly effective MRF optimization algorithms, the field of exploiting their optimization capability with parameter learning to alleviate

each other's drawbacks is rarely explored. A few works provide this capability in certain cases, such as CRFasRNN in semantic segmentation [5] and SGMNet in stereo vision [7], with less effective MRF algorithms, that is MF and SGM respectively. Thus, it is important to adopt highly effective and efficient MRF inference algorithms for optimization and end-to-end learning.

MRF Optimization. Determining an effective MRF optimization algorithm needs a thorough study of the possibility of their optimization capability, differentiability, and time efficiency. In the two main categories of MRF optimization algorithms, namely move-making algorithms (known as graph cuts) [10,11,12,13,14,15] and message passing algorithms [1,16,17,9,18,19,20,21], the state-of-the-art methods are α-expansion [12] and Sequential Tree-Reweighted Message Passing (TRWS) [9] respectively. The move-making algorithms, however, cannot easily be used for parameter learning as they are not differentiable and are usually limited to certain types of energy functions.

In contrast, message passing algorithms adapt better to any energy functions and can be made differentiable and fast if well designed. Some works in probabilistic graphical models indeed demonstrate the learning ability of TRW algorithms with sum-product and max-product [16,20] message passing. A comprehensive study and comparison of these methods can be found in Middlebury [4] and OpenGM [22]. Although SGM [1] is not in the benchmark, it was proved to have a high running efficiency due to the fast one-dimensional Dynamic Programming (DP) that is independent in each scanline and scanning direction [1].

End-to-End Learning. Sum-product TRW [23,24,25] and mean-field [5,26,27] have been used for end-to-end learning for semantic segmentation, which presents their highly effective learning ability. Meanwhile, for stereo vision, several MRF/CRF based methods [7,28,29], such as SGM-related, have been proposed. These further indicate the high efficiency of selected MRF optimization algorithms in end-to-end learning.

In our work, we improve optimization effectiveness and time efficiency based on classical SGM and TRWS. In particular, we revise the standard SGM and make it iterative in order to improve its optimization capability. We denote the resulting algorithm as ISGMR. Our other algorithm, TRWP, is a massively parallelizable version of TRWS, which greatly increases running speed without losing the optimization effectiveness.

3 Message Passing Algorithms

We first briefly review the typical form of a pairwise MRF energy function and discuss two highly parallelizable message passing approaches, ISGMR and TRWP. Such a parallelization capability is essential for fast implementation on GPU and enables relatively straightforward integration to existing deep learning models.

3.1 Pairwise MRF Energy Function

Let X_i be a random variable taking label $x_i \in \mathcal{L}$. A pairwise MRF energy function defined over a set of such variables, parametrized by $\mathbf{\Theta} = \{\theta_i, \theta_{i,j}\}$, is written as

$$E(\mathbf{x} \mid \mathbf{\Theta}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{i,j}(x_i, x_j) \,, \tag{1}$$

where θ_i and $\theta_{i,j}$ denote unary potentials and pairwise potentials respectively, $\mathcal{V}$ is the set of vertices (corresponding, for instance, to image pixels or superpixels), and $\mathcal{E}$ is the set of edges in the MRF (usually encoding a 4-connected or 8-connected grid).

3.2 Iterative Semi-Global Matching Revised

We first introduce the standard SGM for stereo vision supporting only a single iteration. With its connection to message passing, we then revise its message update equation and introduce an iterative version. Figure 1 shows a 4-connected SGM on a grid MRF.

3.2.1 Revised Semi-Global Matching. We cast the popular SGM algorithm [1] as an optimization method for a particular MRF and discuss its relation to message passing as noted in [8]. In SGM, pairwise potentials are simplified for all edges $(i, j) \in \mathcal{E}$ as

$$\theta_{i,j}(\lambda, \mu) = \theta_{i,j}(|\lambda - \mu|) = \begin{cases} 0 & \text{if } \lambda = \mu \ , \\ P_1 & \text{if } |\lambda - \mu| = 1 \ , \\ P_2 & \text{if } |\lambda - \mu| \geq 2 \ , \end{cases} \tag{2}$$

where $0 < P_1 \leq P_2$. The idea of SGM relies on cost aggregation in multiple directions (each direction having multiple one-dimensional scanlines) using Dynamic Programming (DP). The main observation made by [8] is that, in SGM the unary potentials are over-counted $|\mathcal{R}| - 1$ times (where $\mathcal{R}$ denotes the set of directions) compared to the standard message passing and this over-counting corrected SGM is shown to perform slightly better in [30]. Noting this, we use symbol $m_i^r(\lambda)$ to denote the message-vector passed **to** node i, along a scan-line in the direction r, **from** the previous node, denoted $i - r$. This is a vector indexed by $\lambda \in \mathcal{L}$. Now, the SGM update is *revised* from

$$m_i^r(\lambda) = \min_{\mu \in \mathcal{L}} \left(\theta_i(\lambda) + m_{i-r}^r(\mu) + \theta_{i-r,i}(\mu, \lambda) \right) , \tag{3}$$

which is the form given in [1], to

$$m_i^r(\lambda) = \min_{\mu \in \mathcal{L}} \left(\theta_{i-r}(\mu) + m_{i-r}^r(\mu) + \theta_{i-r,i}(\mu, \lambda) \right) . \tag{4}$$

The $m_i^r(\lambda)$ represents the minimum cost due to possible assignments to all nodes previous to node i along the scanline in direction r, and assigning label λ to node i. It does not include the cost $\theta_i(\lambda)$ associated with node i itself.

Since subtracting a fixed value for all λ from messages preserves minima, the message $m_i^r(\lambda)$ can be reparametrized as

$$m_i^r(\lambda) = m_i^r(\lambda) - \min_{\mu \in \mathcal{L}} m_i^r(\mu) \ , \tag{5}$$

which does not alter the minimum energy. Since the values of $\theta_i(\lambda)$ are not included in the messages, the final cost at a particular node i at label λ is *revised* from

$$c_i(\lambda) = \sum_{r \in \mathcal{R}} m_i^r(\lambda) \tag{6}$$

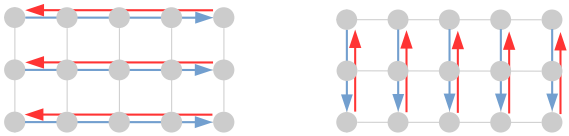


Fig. 1: An example of 4-connected SGM on a grid MRF: left-right, right-left, up-down, down-up. Message passing along all these scanlines can be accomplished in parllel.

to

$$c_i(\lambda) = \theta_i(\lambda) + \sum_{r \in \mathcal{R}} m_i^r(\lambda) \,, \tag{7}$$

which is the sum of messages over all the directions plus the unary term. The final labelling is then obtained by

$$x_i^* = \underset{\lambda \in \mathcal{L}}{\operatorname{argmin}}\, c_i(\lambda) \,, \quad \forall\, i \in \mathcal{V} \,. \tag{8}$$

Here, the message update in the revised SGM, *i.e.*, Eq. (4), is performed in parallel for all scanlines for all directions. This massive parallelization makes it suitable for real-time applications [31] and end-to-end learning for stereo vision [7].

3.2.2 Iteration of Revised Semi-Global Matching. In spite of the revision for the over-counting problem, the 3-penalty pairwise potential in Eq. (2) is insufficient to obtain dominant penalties under a large range of disparities in different camera settings. To this end, we consider more general pairwise potentials $\theta_{i,j}(\lambda, \mu)$ and introduce an iterative version of the revised SGM. The message update for the iterative version is

$$m_i^{r,k+1}(\lambda) = \min_{\mu \in \mathcal{L}} \big(\theta_{i-r}(\mu) + \theta_{i-r,i}(\mu, \lambda) + m_{i-r}^{r,k+1}(\mu) + \sum_{d \in \mathcal{R} \setminus \{r, r^-\}} m_{i-r}^{d,k}(\mu) \big), \tag{9}$$

where r^- denotes the opposite direction of r and $m_{i-r}^{r,k+1}(\mu)$ denotes the updated message in kth iteration while $m_{i-r}^{r,k}(\mu)$ is updated in $(k-1)$th iteration. The exclusion of the messages from direction r^- is important to ensure that the update is analogous to the standard message passing and the same energy function is minimized at each iteration. A simple combination of several standard SGMs does not satisfy this rule and performs worse than our iterative version, as reported in Tables 1-2. Usually, $\mathbf{m}^r$ for all $r \in \mathcal{R}$ are initialized to 0, the exclusion of r^- from $\mathcal{R}$ is thus redundant for a single iteration but not multiple iterations. Even so, messages can be reparametrized by Eq. (5).

After multiple iterations, the final cost for node $i \in \mathcal{V}$ is calculated by Eq. (7), and the final labelling is calculated in the same manner as Eq. (8). We denote this iterative and revised SGM as ISGMR, summarized in Algorithm 1.

In sum, the improvement of ISGMR from SGM lies in the exclusion of over-counted unary terms by Eq. (4) to increase the effects of pairwise terms as well as the iterative energy minimization by Eq. (9) to further decrease the energy with updated messages.

Algorithm 1: Forward Propagation of ISGMR

Input: Energy parameters $\mathbf{\Theta} = \{\theta_i, \theta_{i,j}(\cdot,\cdot)\}$, set of nodes $\mathcal{V}$, edges $\mathcal{E}$, directions $\mathcal{R}$, iteration number K. We replace $m^{r,k}$ by m^r and $m^{r,k+1}$ by $\hat{m}^r$ for simplicity.

Output: Labelling $\mathbf{x}^*$ for optimization, costs $\{c_i(\lambda)\}$ for learning, indices $\{p^r_{k,i}(\lambda)\}$ and $\{q^r_{k,i}\}$ for backpropagation.

$\hat{\mathbf{m}} \leftarrow 0$ and $\mathbf{m} \leftarrow 0$ ▷initialize all messages

for iteration $k \in \{1, \ldots, K\}$ **do**

  forall directions $r \in \mathcal{R}$ **do** ▷**parallel**

    forall scanlines t in direction r **do** ▷**parallel**

      for node i in scanline t **do** ▷**sequential**

        for label $\lambda \in \mathcal{L}$ **do**

          $\Delta(\lambda,\mu) \leftarrow \theta_{i-r}(\mu) + \theta_{i-r,i}(\mu,\lambda) + \hat{m}^r_{i-r}(\mu) + \sum_{d \in \mathcal{R}\setminus\{r,r^-\}} m^d_{i-r}(\mu)$

          $p^r_{k,i}(\lambda) \leftarrow \mu^* \leftarrow \operatorname{argmin}_{\mu\in\mathcal{L}} \Delta(\lambda,\mu)$ ▷store index

          $\hat{m}^r_i(\lambda) \leftarrow \Delta(\lambda,\mu^*)$ ▷message update (9)

        $q^r_{k,i} \leftarrow \lambda^* \leftarrow \operatorname{argmin}_{\lambda\in\mathcal{L}} \hat{m}^r_i(\lambda)$ ▷store index

        $\hat{m}^r_i(\lambda) \leftarrow \hat{m}^r_i(\lambda) - \hat{m}^r_i(\lambda^*)$ ▷reparametrization (5)

  $\mathbf{m} \leftarrow \hat{\mathbf{m}}$ ▷update messages after iteration

$c_i(\lambda) \leftarrow \theta_i(\lambda) + \sum_{r\in\mathcal{R}} m^r_i(\lambda), \forall i \in \mathcal{V}, \lambda \in \mathcal{L}$ ▷Eq. (7)

$x^*_i \leftarrow \operatorname{argmin}_{\lambda\in\mathcal{L}} c_i(\lambda), \forall i \in \mathcal{V}$ ▷Eq. (8)

3.3 Parallel Tree-Reweighted Message Passing

TRWS [9] is another state-of-the-art message passing algorithm that optimizes the Linear Programming (LP) relaxation of a general pairwise MRF energy given in Eq. (1). The main idea of the family of TRW algorithms [32] is to decompose the underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of the MRF with parameters $\mathbf{\Theta}$ into a combination of trees where the sum of parameters of all the trees is equal to that of the MRF, *i.e.*, $\sum_{T\in\mathcal{T}} \mathbf{\Theta}_T = \mathbf{\Theta}$. Then, at each iteration message passing is performed in each of these trees independently, followed by an averaging operation. Even though any combinations of trees would theoretically result in the same final labelling, the best performance is achieved by choosing a monotonic chain decomposition and a sequential message passing update rule, which is TRWS. Interested readers please refer to [9] for more details.

Since we intend to enable fast message passing by exploiting parallelism, our idea is to choose a tree decomposition that can be massively parallelized, denoted as TRWP. In the literature, edge-based or tree-based parallel TRW algorithms have been considered, namely, TRWE and TRWT in the probability space (specifically sum-product message passing) rather than for minimizing the energy [32]. Optimizing in the probability domain involves exponential calculations which are prone to numerical instability, and the sum-product version requires $\mathcal{O}(|\mathcal{R}||\mathcal{L}|)$ times more memory compared to the min-sum message passing in backpropagation. More details are in Appendix E.

Correspondingly, our TRWP directly minimizes the energy in the min-sum message passing fashion similar to TRWS, and thus, its update can be written as

$$m^r_i(\lambda) = \min_{\mu\in\mathcal{L}} \big(\rho_{i-r,i}(\theta_{i-r}(\mu) + \sum_{d\in\mathcal{R}} m^d_{i-r}(\mu)) - m^{r^-}_{i-r}(\mu) + \theta_{i-r,i}(\mu,\lambda)\big). \quad (10)$$

Algorithm 2: Forward Propagation of TRWP

Input: Energy parameters $\boldsymbol{\Theta} = \{\theta_i, \theta_{i,j}(\cdot,\cdot)\}$, set of nodes $\mathcal{V}$, edges $\mathcal{E}$, directions $\mathcal{R}$, tree decomposition coefficients $\{\rho_{i,j}\}$, iteration number K.

Output: Labelling $\mathbf{x}^*$ for optimization, costs $\{c_i(\lambda)\}$ for learning, indices $\{p^r_{k,i}(\lambda)\}$ and $\{q^r_{k,i}\}$ for backpropagation.

$\mathbf{m} \leftarrow 0$ ▷initialize all messages

for iteration $k \in \{1, \dots, K\}$ **do**

  for direction $r \in \mathcal{R}$ **do** ▷**sequential**

    forall scanlines t in direction r **do** ▷**parallel**

      for node i in scanline t **do** ▷**sequential**

        for label $\lambda \in \mathcal{L}$ **do**

          $\Delta(\lambda,\mu) \leftarrow \rho_{i-r,i}\big(\theta_{i-r}(\mu) + \sum_{d\in\mathcal{R}} m^d_{i-r}(\mu)\big) - m^{\bar{r}}_{i-r}(\mu) + \theta_{i-r,i}(\mu,\lambda)$

          $p^r_{k,i}(\lambda) \leftarrow \mu^* \leftarrow \operatorname{argmin}_{\mu\in\mathcal{L}} \Delta(\lambda,\mu)$ ▷store index

          $m^r_i(\lambda) \leftarrow \Delta(\lambda,\mu^*)$ ▷message update (10)

        $q^r_{k,i} \leftarrow \lambda^* \leftarrow \operatorname{argmin}_{\lambda\in\mathcal{L}} m^r_i(\lambda)$ ▷store index

        $m^r_i(\lambda) \leftarrow m^r_i(\lambda) - m^r_i(\lambda^*)$ ▷reparametrization (5)

$c_i(\lambda) \leftarrow \theta_i(\lambda) + \sum_{r\in\mathcal{R}} m^r_i(\lambda), \forall i \in \mathcal{V}, \lambda \in \mathcal{L}$ ▷Eq. (7)

$x^*_i \leftarrow \operatorname{argmin}_{\lambda\in\mathcal{L}} c_i(\lambda), \forall i \in \mathcal{V}$ ▷Eq. (8)

Here, the coefficient $\rho_{i-r,i} = \gamma_{i-r,i}/\gamma_{i-r}$, where $\gamma_{i-r,i}$ and γ_{i-r} are the number of trees containing the edge $(i-r, i)$ and the node $i-r$ respectively in the considered tree decomposition. For loopy belief propagation, since there is no tree decomposition, $\rho_{i-r,i} = 1$. For a 4-connected graph decomposed into all horizontal and vertical one-dimensional trees, we have $\rho_{i-r,i} = 0.5$ for all edges.

Note that, similar to ISGMR, we use the scanline to denote a tree. The above update can be performed in parallel for all scanlines in a single direction; however, the message updates over a scanline are sequential. The same reparametrization Eq. (5) is applied. While TRWP cannot guarantee the non-decreasing monotonicity of the lower bound of energy, it dramatically improves the forward propagation speed and yields virtually similar minimum energies to those of TRWS. The procedure is in Algorithm 2.

In sum, our TRWP benefits from a high speed-up without losing optimization capability by the massive GPU parallelism over individual trees that are decomposed from the single-chain tree in TRWS. All trees in each direction r are paralleled by Eq. (10).

3.4 Relation between ISGMR and TRWP

Both ISGMR and TRWP use messages from neighbouring nodes to perform recursive and iterative message updates via dynamic programming. Comparison of Eq. (9) and Eq. (10) indicates the introduction of the coefficients $\{\rho_{i-r,i}\}$. This is due to the tree decomposition, which is analogous to the difference between loopy belief propagation and TRW algorithms. The most important difference, however, is the way message updates are defined. Specifically, within an iteration, ISGMR can be parallelized over all directions since the most updated messages $\hat{\mathbf{m}}^r$ are used only for the current scanning direction r and previous messages are used for the other directions (refer Eq. (9)). In contrast, aggregated messages in TRWP are up-to-date *direction-by-direction*, which largely contributes to the improved effectiveness of TRWP over ISGMR.

3.5 Fast Implementation by Tree Parallelization

Independent trees make the parallelization possible. We implemented on CPU and GPU, where for the C++ multi-thread versions (CPU), 8 threads on Open Multi-Processing (OpenMP) [33] are used while for the CUDA versions (GPU), 512 threads per block are used. Each tree is headed by its first node by interpolation. The node indexing details for efficient parallelism are provided in Appendix C. In the next section, we derive efficient backpropagation through each of these algorithms for parameter learning.

4 Differentiability of Message Passing

Effective and differentiable MRF optimization algorithms can greatly improve the performance of end-to-end learning. Typical methods such as CRFasRNN for semantic segmentation [5] by MF and SGMNet for stereo vision [7] by SGM use inferior inferences in the optimization capability compared to ISGMR and TRWP.

In order to embed ISGMR and TRWP into end-to-end learning, differentiability of them is required and essential. Below, we describe the gradient updates for the learnable MRF model parameters, and detailed derivations are given in Appendix D. The backpropagation pseudocodes are in Algorithms 3-4 in Appendix A.

Since ISGMR and TRWP use min-sum message passing, no exponent and logarithm are required. Only indices in message minimization and reparametrization are stored in two unsigned 8-bit integer tensors, denoted as $\{p^r_{k,i}(\lambda)\}$ and $\{q^r_{k,i}\}$ with indices of direction r, iteration k, node i, and label λ. This makes the backpropagation time less than 50% of the forward propagation time. In Figure 2a, the gradient updates in backpropagation are performed along edges that have the minimum messages in the forward direction. In Figure 2b, a message gradient at node i is accumulated from all following nodes after i from all backpropagation directions. Below, we denote the gradient of a variable $*$ from loss L as $\nabla * = dL/d*$.

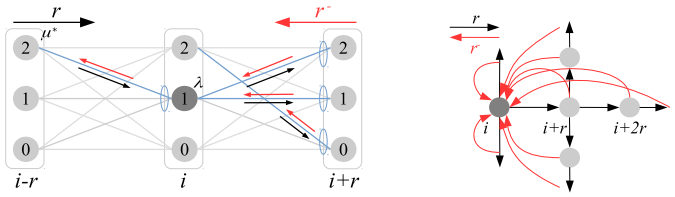
For ISGMR at kth iteration, the gradients of the model parameters in Eq. (9) are

$$\begin{aligned}\nabla\theta_i(\lambda) =& \nabla c_i(\lambda) + \sum_{v\in\mathcal{L}}\sum_{r\in\mathcal{R}}\sum_{\mu\in\mathcal{L}}\Big(\nabla m^{r,k+1}_{i+2r}(\mu)\Big|_{v=p^r_{k,i+2r}(\mu)} \\ &+ \sum_{d\in\mathcal{R}\setminus\{r,r^-\}}\nabla m^{d,k}_{i+r+d}(\mu)\Big|_{v=p^d_{k,i+r+d}(\mu)}\Big)\Big|_{\lambda=p^r_{k,i+r}(v)},\end{aligned} \tag{11}$$

$$\nabla\theta_{i-r,i}(\mu,\lambda) = \nabla m^{r,k+1}_i(\lambda)\Big|_{\mu=p^r_{k,i}(\lambda)}. \tag{12}$$

Importantly, within an iteration in ISGMR, $\nabla\mathbf{m}^{\mathbf{r},k}$ are updated but do not affect $\nabla\mathbf{m}^{\mathbf{r},k+1}$ until the backpropagation along all directions $\mathbf{r}$ is executed (line 18 in Algorithm 3 in Appendix A). This is because within kth iteration, independently updated $\mathbf{m}^{r,k+1}$ in r will not affect $\mathbf{m}^{d,k}, \forall d \in \mathcal{R}\setminus\{r, r^-\}$, until the next iteration (line 12 in Algorithm 1).

In contrast, message gradients in TRWP from a direction will affect messages from other directions since, within an iteration in the forward propagation, message updates are *direction-by-direction*. For TRWP at kth iteration, $\nabla\theta_i(\lambda)$ related to Eq. (10) is



(a) message passing (b) gradient accumulation

Fig. 2: Forward and backward propagation, a target node is in dark gray, r: forward direction, r^-: backpropagation direction. (a) blue ellipse: min operation as MAP, blue line: an edge having the minimum message. (b) a message gradient at node i accumulated from nodes in r^-.

$$\nabla\theta_i(\lambda) = \nabla c_i(\lambda) + \sum_{v\in\mathcal{L}}\sum_{r\in\mathcal{R}}\sum_{\mu\in\mathcal{L}}\Big(-\nabla m_i^{r^-}(\mu)\Big|_{v=p_{k,i}^{r^-}(\mu)} + \sum_{d\in\mathcal{R}} \rho_{i+r,i+r+d}\nabla m_{i+r+d}^{d}(\mu)\big|_{v=p_{k,i+r+d}^{d}(\mu)}\Big)\Big|_{\lambda=p_{k,i+r}^{r}(v)}, \tag{13}$$

where coefficient $\rho_{i+r,i+r+d}$ is for the edge connecting node $i+r$ and its next one in direction d which is denoted as node $i+r+d$, and the calculation of $\nabla\theta_{i-r,i}(\lambda,\mu)$ is in the same manner as Eq. (12) by replacing $m^{r,k+1}$ with m^r.

The backpropagation of TRWP can be derived similarly to ISGMR. We must know that gradients of the unary potentials and the pairwise potentials are accumulated along the opposite direction of the forward scanning direction. Therefore, an updated message is, in fact, a new variable, and its gradient should not be accumulated by its previous value but set to 0. This is extremely important, especially in ISGMR. It requires the message gradients to be accumulated and assigned in every iteration (lines 17-18 in Algorithm 3 in Appendix A) and be zero-out (lines 4 and 16 in Algorithm 3 and line 14 in Algorithm 4 in Appendix A). Meanwhile, gradient derivations of ISGMR and characteristics are provided in Appendix D.

5 Experiments

Below, we evaluated the optimization capability of message passing algorithms on stereo vision and image denoising with fixed yet qualified data terms from benchmark settings. In addition, differentiability was evaluated by end-to-end learning for 21-class semantic segmentation. The experiments include effectiveness and efficiency studies of the message passing algorithms. Additional experiments are in Appendix F.

We implemented SGM, ISGMR, TRWP in C++ with single and multiple threads, PyTorch, and CUDA from scratch. PyTorch versions are for time comparison and gradient checking. For a fair comparison, we adopted benchmark code of TRWS from [34] with general pairwise functions; MF followed Eq. (4) in [6]. For iterative SGM, unary potentials were reparametrized by Eq. (6). OpenGM [22] can be used for more comparisons in optimization noting TRWS as one of the most effective inference methods.

Our experiments were on 3.6GHz i7-7700 Intel(R) Core(TM) and Tesla P100 SXM2.

5.1 Optimization for Stereo Vision and Image Denoising

The capability of minimizing an energy function determines the significance of selected algorithms. We compared our ISGMR and TRWP with MF, SGM with single and multiple iterations, and TRWS. The evaluated energies are calculated with 4 connections.

Datasets. For stereo vision, we used Tsukuba, Teddy, Venus, Map, and Cones from Middlebury [35,36], 000041_10 and 000119_10 from KITTI2015 [37,38], and delivery_area_1l and facade_1 from ETH3D two-view [39] for different types of stereo views. For image denoising, Penguin and House from Middlebury dataset [3] were used.

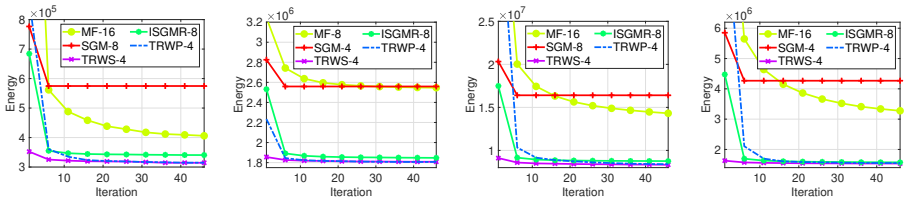
MRF model parameters. Model parameters include unary and pairwise potentials. In practice, the pairwise potentials consist of a pairwise function and edge weights, as $\theta_{i,j}(\lambda,\mu) = \theta_{i,j}V(\lambda,\mu)$. For the pairwise function $V(\cdot,\cdot)$, one can adopt (truncated) linear, (truncated) quadratic, Cauchy, Huber, *etc.*, [40]. For the edge weights $\theta_{i,j}$, some methods apply a higher penalty on edge gradients under a given threshold. We set it as a constant for the comparison with SGM. Moreover, we adopted edge weights in [34] and pairwise functions for Tsukuba, Teddy, and Venus, and [11] for Cones and Map; for the others, the pairwise function was linear and edges weights were 10. More evaluations with constant edge weights are given in Appendix F.

Number of directions matters. In Figure 3, ISGMR-8 and TRWP-4 outperform the others in ISGMR-related and TRWP-related methods in most cases. From the experiments, 4 directions are sufficient for TRWP, but for ISGMR energies with 8 directions are lower than those with 4 directions. This is because messages from 4 directions in ISGMR are insufficient to gather local information due to independent message updates in each direction. In contrast, messages from 4 directions in TRWP are highly updated in each direction and affected by those from the other directions. Note that in Eq. (7) messages from all directions are summed equally, this makes the labels by TRWP oversmooth within the connection area, for example, the camera is oversmooth in Figure 4n. Overall, TRWP-4 and ISGMR-8 are the best.

ISGMR vs SGM. [30] demonstrates the decrease in energy of the over-count corrected SGM compared with the standard SGM. The result shows the improved optimization results achieved by subtracting unary potentials $(|\mathcal{R}|-1)$ times. For experimental completion, we show both the decreased energies and improved disparity maps produced by ISGMR. From Tables 1-2, SGM-related energies are much higher than ISGMR's because of the over-counted unary potentials. Moreover, ISGMR at the 50th iteration has much a lower energy value than the 1st iteration, indicating the importance of iterations, and is also much lower than those for MF and SGM at the 50th iteration.

TRWP vs TRWS. TRWP and TRWS have the same manner of updating messages and could have similar minimum energies. Generally, TRWS has the lowest energy; at the 50th iteration, however, TRWP-4 has lower energies, for instance, Tsukuba and Teddy in Table 1 and Penguin and House in Table 2. For TRWP, 50 iterations are sufficient to show its high optimization capability, as shown in Figure 3. More visualizations of Penguin and House denoising are in Appendix F.

[3] http://vision.middlebury.edu/MRF/results



(a) Tsukuba (b) Teddy (c) 000002_11 (d) delivery_area_1l

Fig. 3: Convergence with the connections having the minimum energy in Table 1.

Table 1: Energy minimization for stereo vision. ISGMR is better than SGM and TRWP obtains similar energies as TRWS. ISGMR and TRWP outperform MF and SGM.

Method	Tsukuba		Teddy		000002_11		delivery_area_1l	
	1 iter	50 iter	1 iter	50 iter	1 iter	50 iter	1 iter	50 iter
MF-4	3121704	1620524	3206347	2583784	82523536	44410056	19945352	9013862
SGM-4	873777	644840	2825535	2559016	24343250	18060026	5851489	4267990
TRWS-4	352178	314393	1855625	1807423	9109976	8322635	1628879	1534961
ISGMR-4 (ours)	824694	637996	2626648	1898641	22259606	12659612	5282024	2212106
TRWP-4 (ours)	869363	314037	2234163	1806990	40473776	8385450	9899787	1546795
MF-8	2322139	504815	3244710	2545226	61157072	18416536	16581587	4510834
SGM-8	776706	574758	2868131	2728682	20324684	16406781	5396353	4428411
ISGMR-8 (ours)	684185	340347	2532071	1847833	17489158	8753990	4474404	1571528
TRWP-8 (ours)	496727	348447	1981582	1849287	18424062	8860552	4443931	1587917
MF-16	1979155	404404	3315900	2622047	46614232	14192750	13223338	3229021
SGM-16	710727	587376	2907051	2846133	18893122	16791762	5092094	4611821
ISGMR-16 (ours)	591554	377427	2453592	1956343	15455787	9556611	3689863	1594877
TRWP-16 (ours)	402033	396036	1935791	1976839	11239113	9736704	2261402	1630973

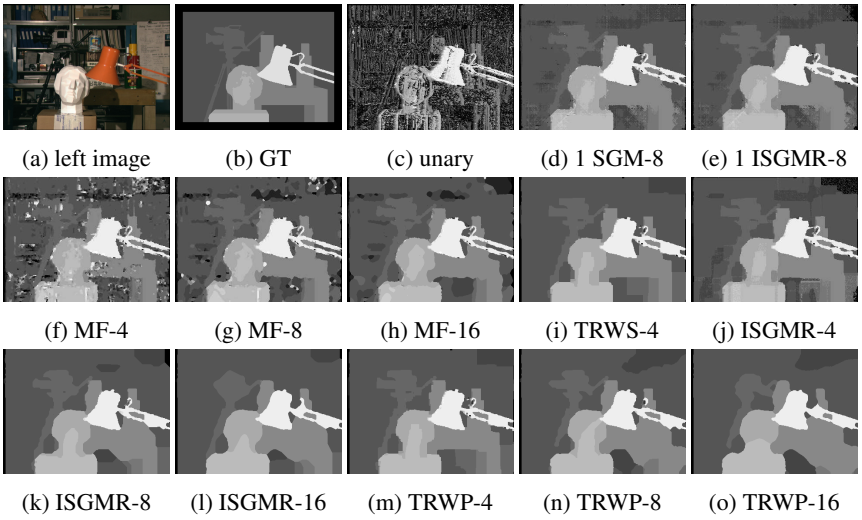
5.2 End-to-End Learning for Semantic Segmentation

Although deep network and multi-scale strategy on CNN make semantic segmentation smooth and continuous on object regions, effective message passing inference on pairwise MRFs is beneficial for fine results with auxiliary edge information. The popular denseCRF [6] demonstrated the effectiveness of using MF inference and the so-called dense connections; our experiments, however, illustrated that with local connections, superior inferences, such as TRWS, ISGMR, and TRWP, have a better convergence ability than MF and SGM to improve the performance.

Below, we adopted TRWP-4 and ISGMR-8 as our inference methods and negative logits from DeepLabV3+ [41] as unary terms. Edge weights from Canny edges are in the form of $\theta_{ij} = 1 - |e_i - e_j|$, where e_i is a binary Canny edge value at node i. Potts model was used for pairwise function $V(\lambda, \mu)$. Since MF required much larger GPU memory than others due to its dense gradients, for practical purposes we used MF-4 for learning with the same batch size 12 within our GPU memory capacity.

Datasets. We used PASCAL VOC 2012 [42] and Berkeley benchmark [43], with 1449 samples of the PASCAL VOC 2012 val set for validation and the other 10582 for training. These datasets identify 21 classes with 20 objects and 1 background.

CNN learning parameters. We trained the state-of-the-art DeepLabV3+ (ResNet101 as the backbone) with initial learning rate 0.007, "poly" learning rate decay scheduler,



(a) left image (b) GT (c) unary (d) 1 SGM-8 (e) 1 ISGMR-8

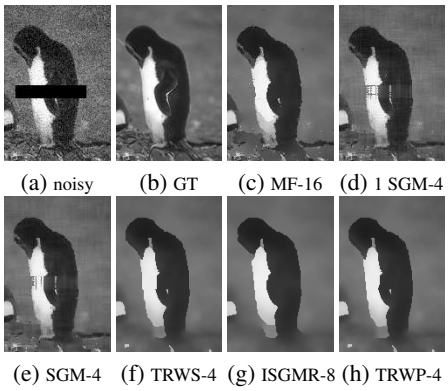
(f) MF-4 (g) MF-8 (h) MF-16 (i) TRWS-4 (j) ISGMR-4

(k) ISGMR-8 (l) ISGMR-16 (m) TRWP-4 (n) TRWP-8 (o) TRWP-16

Fig. 4: Disparities of Tsukuba. (d)-(e) are at 1st iteration. (f)-(o) are at 50th iteration. (j) and (l) have the lowest energies in ISGMR-related and TRWP-related methods respectively. TRWP-4 and TRWS-4 have similar disparities for the most parts.

Table 2: Energy minimization for image denoising at 50th iteration with 4, 8, 16 connections (all numbers divided by 10^3). Our ISGMR or TRWP performs best.

Method	Penguin	House
MF-4	46808	50503
SGM-4	31204	66324
TRWS-4	15361	37572
ISGMR-4 (ours)	16514	37603
TRWP-4 (ours)	15358	37552
MF-8	21956	47831
SGM-8	37520	76079
ISGMR-8 (ours)	15899	39975
TRWP-8 (ours)	16130	40209
MF-16	20742	55513
SGM-16	47028	87457
ISGMR-16 (ours)	17035	46997
TRWP-16 (ours)	17516	47825



(a) noisy (b) GT (c) MF-16 (d) 1 SGM-4

(e) SGM-4 (f) TRWS-4 (g) ISGMR-8 (h) TRWP-4

Fig. 5: Penguin denoising corresponding to the minimum energies marked with gray color in Table 2. ISGMR-8 and TRWP-4 are our proposals.

and image size 512×512. Negative logits from DeepLabV3+ served as unary terms, the learning rate was decreased for learning message passing inference with 5 iterations, *i.e.*, 1e-4 for TRWP and SGM and 1e-6 for ISGMR and MF. Note that we experimented with all of these learning rates for involved inferences and selected the best for demonstration, for instance, for MF the accuracy by 1e-6 is much higher than the one by 1e-4.

In Table 3, ISGMR-8 and TRWP-4 outperform the baseline DeepLabV3+ [41], SGM-8 [1], and MF-4 [6]. Semantic segmentation by ISGMR-8 and TRWP-4 are more

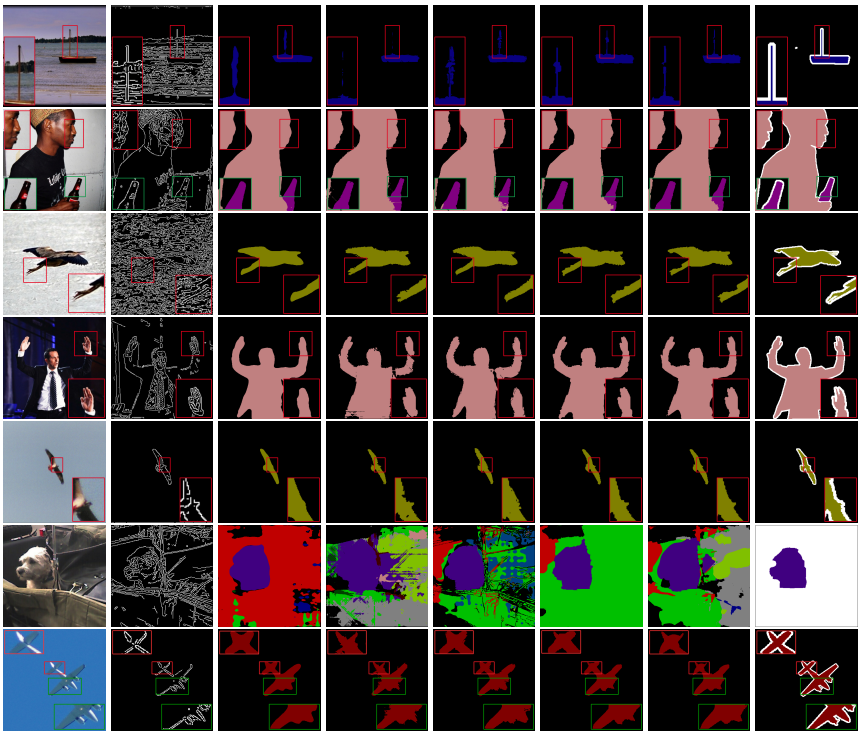
Table 3: Learning for semantic segmentation with mIoU on PASCAL VOC2012 val set.

(a) term weight for TRWP-4

Method	λ	mIoU (%)
+TRWP-4	1	79.27
+TRWP-4	10	79.53
+TRWP-4	20	79.65
+TRWP-4	30	79.44
+TRWP-4	40	79.60

(b) full comparison

Method	λ	mIoU (%)
DeepLabV3+ [41]	-	78.52
+SGM-8 [1]	5	78.94
+MF-4 [6]	5	77.89
+ISGMR-8 (ours)	5	78.95
+TRWP-4 (ours)	20	79.65



(a) RGB (b) Canny (c) baseline (d) SGM-8 (e) MF-4 (f) ISGMR-8 (g) TRWP-4 (h) GT

Fig. 6: Semantic segmentation on PASCAL VOC2012 val set. Last two rows are failure cases due to poor unary terms and missing edges. ISGMR-8 and TRWP-4 are ours.

sharp, accurate, and aligned with the Canny edges and ground-truth (GT) edges, shown in white, than the other inference methods, such as SGM-8 and MF-4 (see Figure 6).

5.3 Speed Improvement

Speed-up by parallelized message passing on a GPU enables a fast inference and end-to-end learning. To be clear, we compared forward and backward propagation times for different implementations using 256×512 size images with 32 and 96 labels.

Table 4: Forward propagation time with 32 and 96 labels. Our CUDA version is averaged over 1000 trials; others over 100 trials. Our CUDA version is 7–32 times faster than PyTorch GPU version. C++ versions are with a single and 8 threads. Unit: second.

Method	PyTorch CPU		PyTorch GPU		C++ single		C++ multiple		CUDA (ours)		Speed-up PyT/CUDA	
	32	96	32	96	32	96	32	96	32	96	32	96
TRWS-4	-	-	-	-	1.95	13.30	-	-	-	-	-	-
ISGMR-4	1.43	11.70	0.96	1.13	3.23	25.19	0.88	5.28	0.03	0.15	32×	8×
ISGMR-8	3.18	24.78	1.59	1.98	8.25	71.35	2.12	15.90	0.07	0.27	23×	7×
ISGMR-16	7.89	52.76	2.34	4.96	30.76	273.68	7.70	62.72	0.13	0.53	18×	9×
TRWP-4	1.40	11.74	0.87	1.08	1.84	15.41	0.76	4.46	0.03	0.15	29×	7×
TRWP-8	3.19	24.28	1.57	1.98	6.34	57.25	1.88	14.22	0.07	0.27	22×	7×
TRWP-16	7.86	51.85	2.82	5.08	28.93	262.28	7.41	60.45	0.13	0.52	22×	10×

Method	PyTorch GPU		CUDA (ours)		Speed-up PyT/CUDA	
	32	96	32	96	32	96
ISGMR-4	7.38	21.48	0.01	0.03	738×	716×
ISGMR-8	18.88	55.92	0.02	0.07	944×	799×
ISGMR-16	58.23	173.02	0.06	0.18	971×	961×
TRWP-4	7.35	21.45	0.01	0.02	735×	1073×
TRWP-8	18.86	55.94	0.02	0.06	943×	932×
TRWP-16	58.26	172.95	0.06	0.16	971×	1081×

Table 5: Backpropagation time. PyTorch GPU is averaged on 10 trials and CUDA on 1000 trials. Ours is 716–1081 times faster than PyTorch GPU. Unit: second.

Forward propagation time. In Table 4, the forward propagation by CUDA implementation is the fastest. Our CUDA versions of ISGMR-8 and TRWP-4 are at least 24 and 7 times faster than PyTorch GPU versions at 32 and 96 labels respectively. In PyTorch GPU versions, we used tensor-wise tree parallelization to highly speed it up for a fair comparison. Obviously, GPU versions are much faster than CPU versions.

Backpropagation time. In Table 5, the backpropagation time clearly distinguishes the higher efficiency of CUDA versions than PyTorch GPU versions. On average, the CUDA versions are at least 700 times faster than PyTorch GPU versions, and only a low memory is used to store indices for backpropagation. This makes the backpropagation much faster than the forward propagation and enables its feasibility in deep learning. Analysis of PyTorch GPU version and our CUDA implementation are in Appendix D.4.

6 Conclusion

In this paper, we introduce two fast and differentiable message passing algorithms, namely, ISGMR and TRWP. While ISGMR improved the effectiveness of SGM, TRWP sped up TRWS by two orders of magnitude without loss of solution quality. Besides, our CUDA implementations achieved at least 7 times and 700 times speed-up compared to PyTorch GPU versions in the forward and backward propagation respectively. These enable end-to-end learning with effective and efficient MRF optimization algorithms. Experiments of stereo vision and image denoising as well as end-to-end learning for semantic segmentation validated the effectiveness and efficiency of our proposals.

Acknowledgement: We would like to thank our colleagues Dylan Campbell and Yao Lu for the discussion of CUDA programming. This work is supported by the Australian Centre for Robotic Vision (CE140100016) and Data61, CSIRO, Canberra, Australia.

References

1. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. TPAMI (2008)
2. Boykov, Y., Jolly, M.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. ICCV (2011)
3. Hassner, M., Sklansky, J.: The use of Markov random fields as models of texture. Computer Graphics and Image Processing (1980)
4. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. TPAMI (2008)
5. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. CVPR (2015)
6. Krähenbühl, P., Koltunz, V.: Efficient inference in fully connected CRFs with gaussian edge potentials. NeurIPS (2011)
7. Seki, A., Pollefeys, M.: SGM-nets: Semi-global matching with neural networks. CVPR (2017)
8. Drory, A., Haubold, C., Avidan, S., Hamprecht, F.A.: Semi-global matching: a principled derivation in terms of message passing. Pattern Recognition (2014)
9. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. TPAMI (2006)
10. Ajanthan, T., Hartley, R., Salzmann, M.: Memory efficient max-flow for multi-label submodular MRFs. CVPR (2016)
11. Ajanthan, T., Hartley, R., Salzmann, M., Li, H.: Iteratively reweighted graph cut for multi-label MRFs with non-convex priors. CVPR (2015)
12. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. TPAMI (2001)
13. Carr, P., Hartley, R.: Solving multilabel graph cut problems with multilabel swap. DICTA (2009)
14. Hartley, R., Ajanthan, T.: Generalized range moves. arXiv:1811.09171 (2018)
15. Veksler, O.: Multi-label moves for MRFs with truncated convex priors. IJCV (2012)
16. Jordan, M.: Learning in graphical models. MIT Press (1998)
17. Kwon, D., Lee, K., Yun, I., Lee, S.: Solving MRFs with higher-order smoothness priors using hierarchical gradient nodes. ACCV (2010)
18. Murphy, K., Weiss, Y., Jordan, M.: Loopy belief propagation for approximate inference: an empirical study. UAI (1999)
19. Pearl, J.: Probabilistic reasoning in intelligent systems. Morgan Kaufmann (1988)
20. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning (2008)
21. Wang, Z., Zhang, Z., Geng, N.: A message passing algorithm for MRF inference with unknown graphs and its applications. ACCV (2014)
22. Kappes, J., Andres, B., Hamprecht, F., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kroeger, T., Kausler, B., Lellmann, J., Savchynskyy, B., Komodakis, N., Rother, C.: A comparative study of modern inference techniques for discrete energy minimization problems. CVPR (2013)
23. Domke, J.: Learning graphical model parameters with approximate marginal inference. TPAMI (2013)
24. Taskar, B., Guestrin, C., Koller, D.: Max-margin Markov networks. MIT Press (2003)
25. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. JMLR (2005)

26. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. ICCV (2015)
27. Lin, G., Shen, C., Hengel, A., Reid, I.: Efficient piecewise training of deep structured models for semantic segmentation. CVPR (2016)
28. Zhang, F., Prisacariu, V., Yang, R., Torr, P.H.: GA-Net: Guided aggregation net for end-to-end stereo matching. CVPR (2019)
29. Knobelreiter, P., Reinbacher, C., Shekhovtsov, A., Pock, T.: End-to-end training of hybrid CNN-CRF models for stereo. CVPR (2017)
30. Facciolo, G., Franchis, C., Meinhardt, E.: MGM: A significantly more global matching for stereo vision. BMVC (2015)
31. Hernandez-Juare, D., Chacon, A., Espinosa, A., Vazquez, D., Moure, J., opez, A.M.L.: Embedded real-time stereo estimation via semi-global matching on the GPU. International Conference on Computational Sciences (2016)
32. Wainwright, M., Jaakkola, T., Willsky, A.: MAP estimation via agreement on (hyper) trees: Message-passing and linear-programming approaches. Transactions on Information Theory (2005)
33. Dagum, L., Menon, R.: OpenMP: an industry standard API for shared-memory programming. Computational Science and Engineering (1998)
34. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. TPAMI (2008)
35. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. IJCV (2002)
36. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. CVPR (2003)
37. Menze, M., Heipke, C., Geiger, A.: Object scene flow. ISPRS Journal of Photogrammetry and Remote Sensing (2018)
38. Menze, M., Heipke, C., Geiger, A.: Joint 3D estimation of vehicles and scene flow. ISPRS Workshop on Image Sequence Analysis (2015)
39. Schops, T., Schonberger, J., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A.: A multi-view stereo benchmark with high-resolution images and multi-camera videos. CVPR (2017)
40. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
41. Chen, L., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. ECCV (2018)
42. Everingham, M., Eslami, S., Gool, L., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes challenge a retrospective. IJCV (2014)
43. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. ICCV (2011)