ECCV 2020

Fast and Differentiable Message Passing on Pairwise Markov Random Fields

(Oral Presentation)

Zhiwei Xu[1,2], Thalaiyasingam Ajanthan[1], Richard Hartley[1]

[1] Australian National University (ANU) and Australian Centre for Robotic Vision (ACRV)
[2] Data61, CSIRO, Canberra, Australia



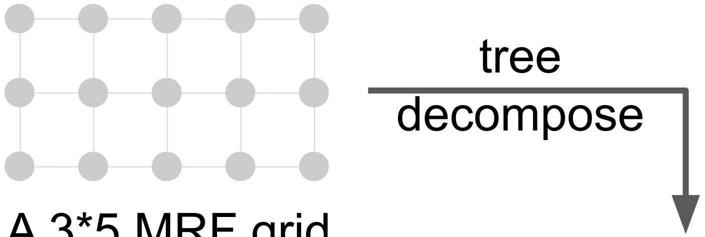
Problem

Limited Spread of **MRF** Optimization Algorithms in Deep Learning due to

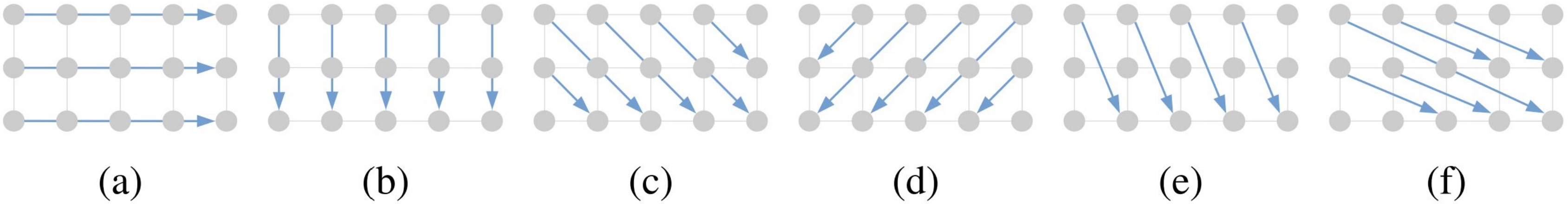
- Hand-crafted model parameters
- Inferior optimization capability
- Non-differentiability
- Low computational efficiency

Background

Markov Random Fields - 2D grids



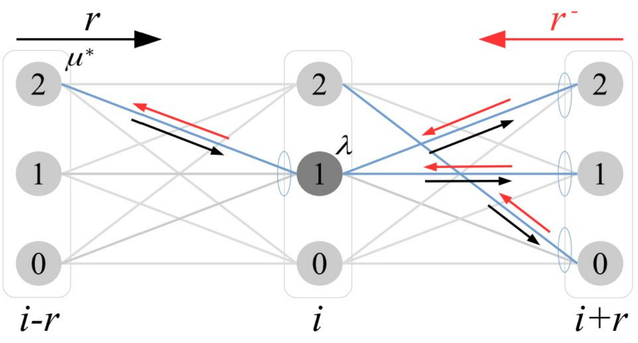
A 3*5 MRF grid



Tree decomposition with scanning directions (6-direction example)

Background

Max-A-Posteriori (MAP) message passing



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

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

MAP

Note: This is in contrast to marginal form, sum() or prod(), in Probabilistic Graph Model (PGM).

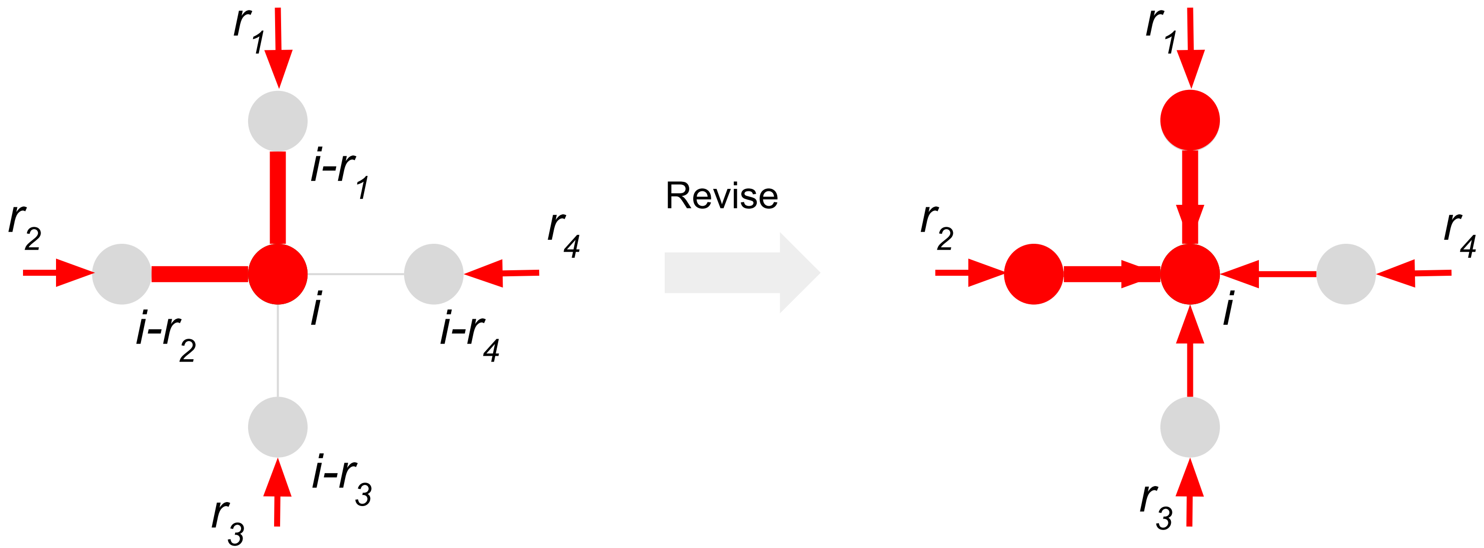
Two Solutions (ISGMR & TRWP)

Solution 1: Iterative Semi-Global Matching Revised (ISGMR)

- Revision of SGM
- Iterative Energy Minimization
- Massive Parallelism over Trees and Directions

Two Solutions (ISGMR & TRWP)

ISGMR: from SGM to our revision



$m^{r_1}+m^{r_2}+m^{r_3}+m^{r_4}=c$ $m^{r_1}+m^{r_2}+m^{r_3}+m^{r_4}+\theta_i=c$

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

Two Solutions (ISGMR & TRWP)

ISGMR: message updates and iteration

Parallel Direction

Message Revision

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

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

Message Aggregation

SGM: $c_i(\lambda) = \sum_{r \in \mathcal{R}} m_i^r(\lambda)$ ➡ Ours: $c_i(\lambda) = \theta_i(\lambda) + \sum_{r \in \mathcal{R}} m_i^r(\lambda)$

NO Over-Counted Unary Terms

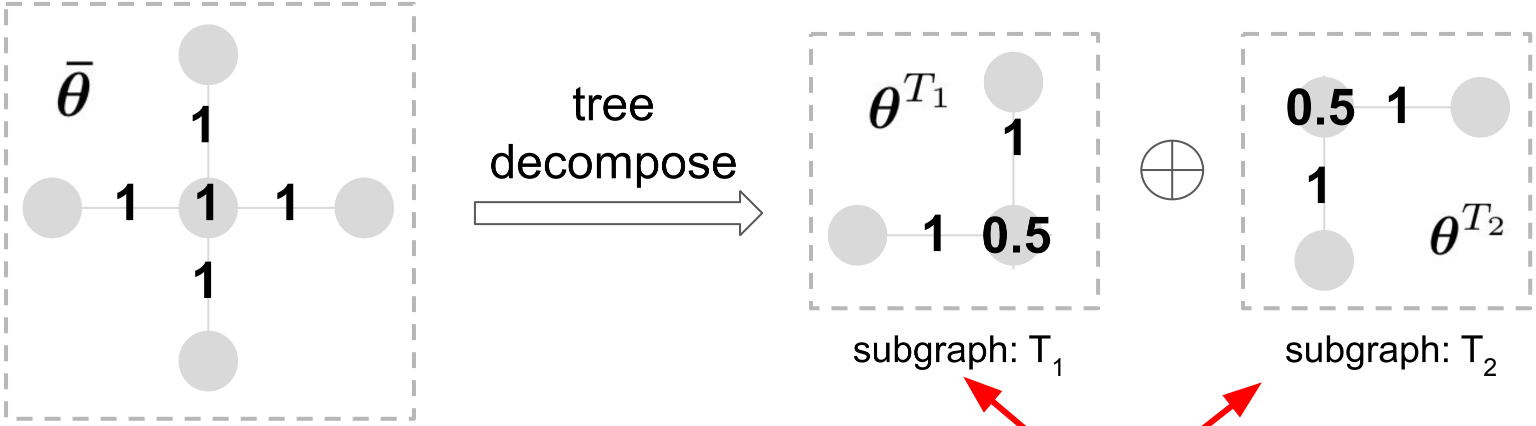
Iteration

Iterative Optimization

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

Two Solutions (ISGMR & TRWP)

Introduction of TRWS: a SOTA energy minimization algorithm



$$\max_{\boxed{\theta}\in\mathcal{A},\sum_T \rho^T\theta^T=\bar{\theta}} \boxed{\sum_T \rho^T \boxed{\min_{\mathbf{x}\in\mathcal{X}} \langle \theta^T, \phi(\mathbf{x})\rangle}}$$

Reparameterization on a larger set

Energy minimization on each subgraph

A lower bound on the condition of $\sum_T \rho^T\theta^T = \bar{\theta}$

Reference: *Convergent Tree-reweighted Message Passing for Energy Minimization*, PAMI, 2006

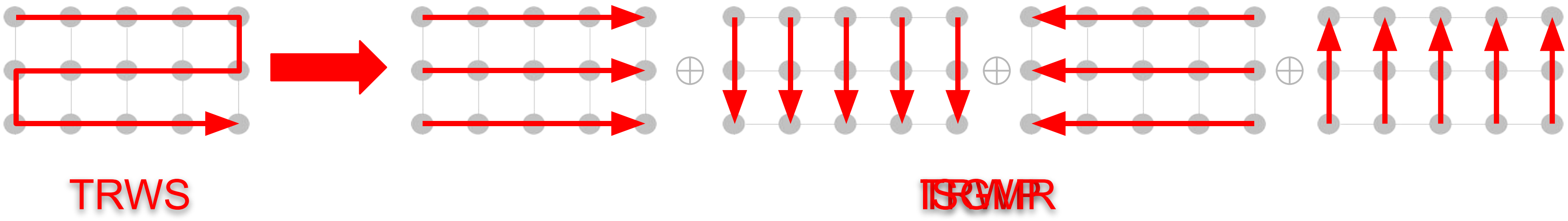
Two Solutions (ISGMR & TRWP)

Solution 2: Parallel Tree-Reweighted Message Passing (TRWP)

- Breaking into Individual Trees from a Single-Chain Tree of TRWS
- Speed-up by Parallelism
- Maintaining High Optimization Ability

Sequential Direction

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

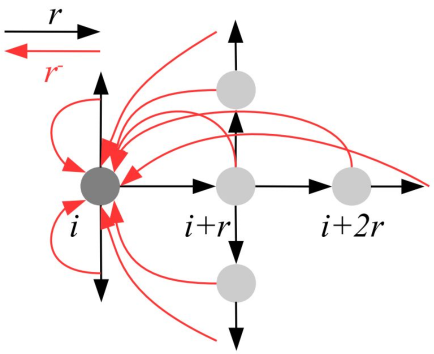


Differentiability

The differentiability involves the accumulated gradients of

- Messages
- Unary potentials
- Pairwise potentials (optional)

Achieved by unrolling the message updates in the forward propagation.



Gradient accumulation at node *i*

Differentiability

Algorithm 3: Backpropagation of ISGMR

Input: Partial energy parameters $\{\theta_{i,j}\}$, gradients of final costs $\nabla \mathbf{c} = \{\nabla c_i(\lambda)\}$, set of nodes $\mathcal{V}$, edges $\mathcal{E}$, directions $\mathcal{R}$, indices $\{p^r_{k,i}(\lambda)\}$, $\{q^r_{k,i}\}$, iteration number K. We replace $\nabla m^{r,k+1}$ by $\nabla \hat{m}^r$ and $\nabla m^{r,k}$ by ∇m^r for simplicity.

Output: Gradients $\{\nabla\theta_i, \nabla\theta_{i,j}(\cdot,\cdot)\}$.

$\nabla \mathbf{m^r} \leftarrow \nabla \mathbf{\Theta_i} \leftarrow \nabla \mathbf{c}$, $\nabla \mathbf{\Theta_{i,j}} \leftarrow 0$ ▷back Eq. (7)
$\nabla \mathbf{\hat{m}^r} \leftarrow \nabla \mathbf{m^r}$ ▷back message updates
for iteration $k \in \{K, ..., 1\}$ **do**
  $\boxed{\nabla \mathbf{m^r} \leftarrow 0}$ ▷zero-out
  forall directions $r \in \mathcal{R}$ **do** ▷**parallel**
    forall scanlines t in direction r **do** ▷**parallel**
      for node i in scanline t **do** ▷**sequential**
        $\lambda^* \leftarrow q^r_{k,i} \in \mathcal{L}$ ▷extract index
        $\nabla \hat{m}^r_i(\lambda^*) -= \sum_{\lambda \in \mathcal{L}} \nabla \hat{m}^r_i(\lambda)$ ▷back Eq. (5)
        for label $\lambda \in \mathcal{L}$ **do**
          $\mu^* \leftarrow p^r_{k,i}(\lambda) \in \mathcal{L}$ ▷extract index
          $\nabla\theta_{i-r}(\mu^*) += \nabla \hat{m}^r_i(\lambda)$ ▷back Eq. (9)
          $\nabla \hat{m}^r_{i-r}(\mu^*) += \nabla \hat{m}^r_i(\lambda)$
          $\nabla m^d_{i-r}(\mu^*) += \nabla \hat{m}^r_i(\lambda), \forall d \in \mathcal{R} \setminus \{r, r^-\}$
          $\nabla\theta_{i-r,i}(\mu^*, \lambda) += \nabla \hat{m}^r_i(\lambda)$
    $\boxed{\nabla \mathbf{\hat{m}}^r \leftarrow 0}$ ▷zero-out
  $\nabla \mathbf{m^r} += \nabla \mathbf{\hat{m}^r}$ ▷gather history gradients
  $\nabla \mathbf{\hat{m}^r} \leftarrow \nabla \mathbf{m^r}$ ▷back message updates after iteration

Algorithm 4: Backpropagation of TRWP

Input: Partial energy parameters $\{\theta_{i,j}\}$, gradients of final costs $\nabla \mathbf{c} = \{\nabla c_i(\lambda)\}$, tree decomposition coefficients $\{\rho_{i,j}\}$, set of nodes $\mathcal{V}$, edges $\mathcal{E}$, directions $\mathcal{R}$, indices $\{p^r_{k,i}(\lambda)\}$, $\{q^r_{k,i}\}$, iteration number K.

Output: Gradients $\{\nabla\theta_i, \nabla\theta_{i,j}(\cdot,\cdot)\}$.

$\nabla \mathbf{m^r} \leftarrow \nabla \mathbf{\Theta_i} \leftarrow d\mathbf{c}$, $d\mathbf{\Theta_{i,j}} \leftarrow 0$ ▷back Eq. (7)
for iteration $k \in \{K, ..., 1\}$ **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**
        $\lambda^* \leftarrow q^r_{k,i} \in \mathcal{L}$ ▷extract index
        $\nabla m^r_i(\lambda^*) -= \sum_{\lambda \in \mathcal{L}} \nabla m^r_i(\lambda)$ ▷back Eq. (5)
        for label $\lambda \in \mathcal{L}$ **do**
          $\mu^* \leftarrow p^r_{k,i}(\lambda) \in \mathcal{L}$ ▷extract index
          $\nabla\theta_{i-r}(\mu^*) += \rho_{i-r,i} \nabla m^r_i(\lambda)$ ▷back Eq. (11)
          $\nabla m^d_{i-r}(\mu^*) += \rho_{i-r,i} \nabla m^r_i(\lambda), \forall d \in \mathcal{R}$
          $\nabla m^{r^-}_{i-r}(\mu^*) -= \nabla m^r_i(\lambda)$
          $\nabla\theta_{i-r,i}(\mu^*, \lambda) += \nabla m^r_i(\lambda)$
    $\boxed{\nabla \mathbf{m}^r \leftarrow 0}$ ▷zero-out

Our Results

- **Effectiveness** of Optimization

  Stereo vision and image denoising

- **Efficiency** of Running Time

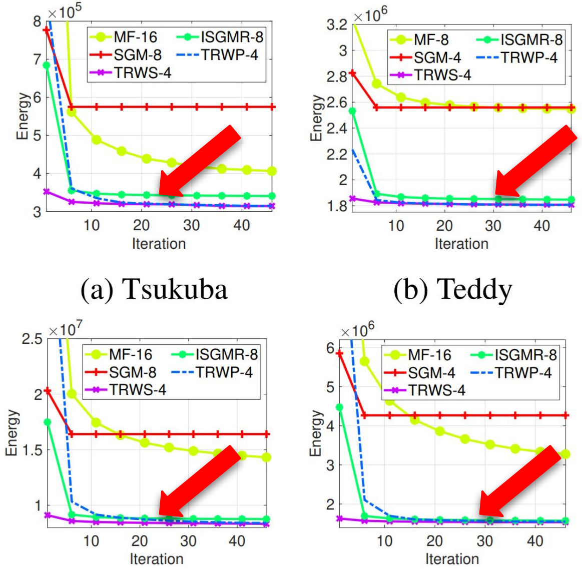
  Forward and backward propagation time

- **Evaluation** on Deep Learning

  Semantic segmentation on PASCAL VOC 2012

Effectiveness (Optimization-Energy Minimization)

- Task: Stereo Vision



(a) Tsukuba (b) Teddy

(c) 000002_11 (d) delivery_area_11

Datasets: Middlebury KITTI2015 ETH3D

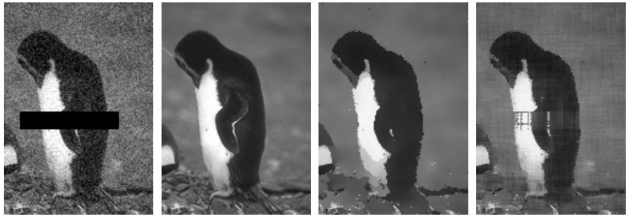
Method	Tsukuba		Teddy		000002_11		delivery_area_1l	
	1 iter	50 iter	1 iter	50 iter	1 iter	50 iter	1 iter	50 iter
MF-4	3121704	1620524	3206347	2583784	82523536	44410056	19945352	9013862
SGM-4	873777	644840	2825535	2559016	24343250	18060026	5851489	4267990
TRWS-4	352178	314393	1855625	1807423	9109976	8322635	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

Method with the lowest energy is the best

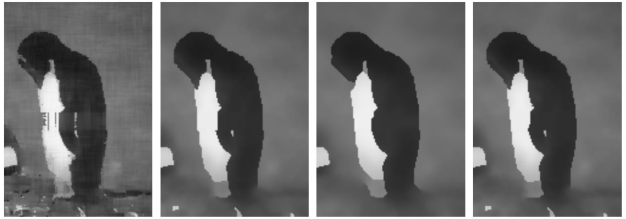
Effectiveness (Optimization-Energy Minimization)

- Task: Image Denoising

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

Efficiency (Computational Complexity)

- Min-Sum Form Message Passing

(a) Forward Pass Time

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×

least

(b) Backpropagation Time

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×

least

Evaluation (Deep Learning)

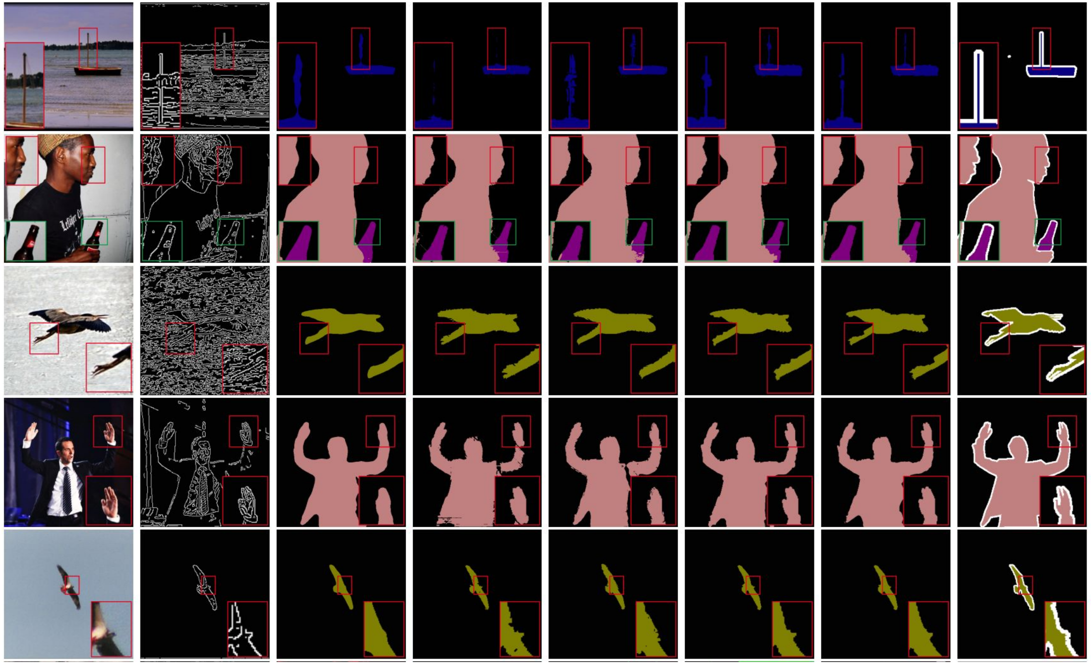
- Task: Semantic Segmentation (PASCAL VOC 2012 Dataset)

(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+ [40]	-	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

Summary

We introduce two message passing algorithms, ISGMR and TRWP, which are

- Fast and Achievable on GPU
- Differentiable
- For Optimization
- For Deep Learning

Code of Our MPLayers:

https://github.com/zwxu064/MPLayers.git

-- End --