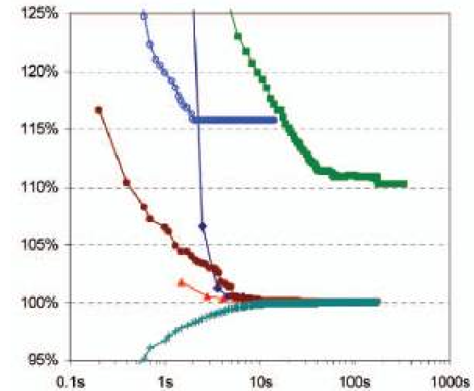
Part 2: Regularizing the estimates: labeling optimization (40 mins)

- **Local labeling optimization**
  - Local data aggregation and cost volume filtering, PAMI 2013
  - PatchMatch Filter, CVPR 2013

- **Global labeling optimization**
  - PatchMatch Belief Propagation, IJCV 2014
  - Sped-up PatchMatch Belief Propagation, ICCV 2015

MRF-based global labeling optimization

- Elegant formulation as Markov random fields



MRF Minimization Results • Code

Richard Szeliski • Ramin Zabih • Daniel Scharstein • Olga Veksler •
Vladimir Kolmogorov • Aseem Agarwala • Marshall Tappen • Carsten Rother

This site contains the results (plots and images) and code accompanying our paper

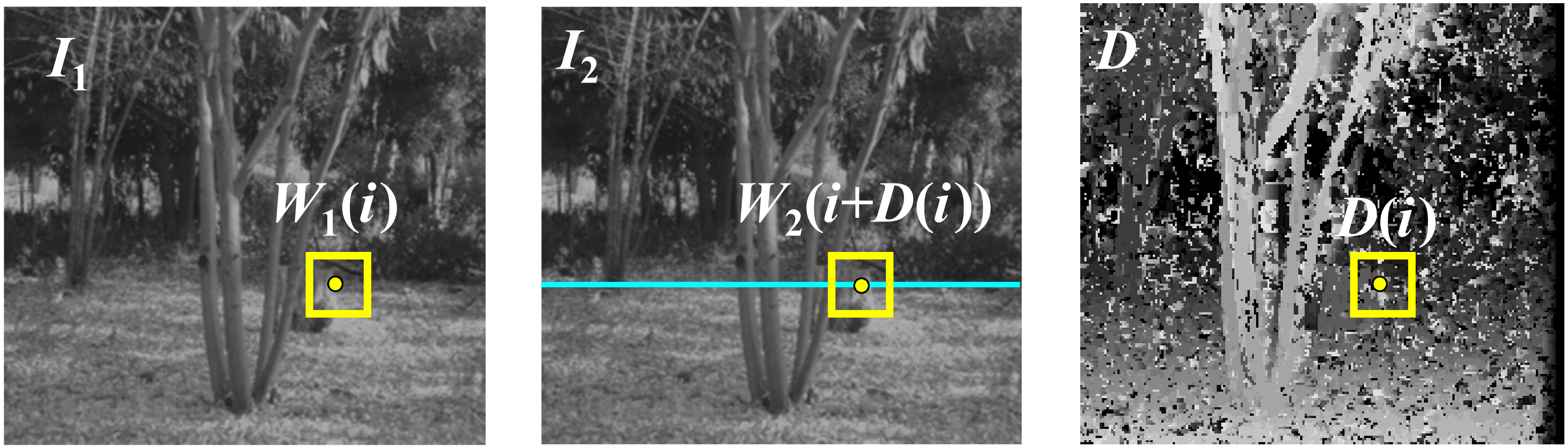
A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors,
IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 30(6):1068-1080, June 2008.

The first version, corresponding to our ECCV 06 paper, is still available here.

$$E = E_d + \lambda E_s$$

- Slow even with efficient energy minimization algorithms

MRF-based global labeling optimization



$$E(D) = \underbrace{\sum_i \left(W_1(i) - W_2(i + D(i))\right)^2}_{\textit{data term}} + \lambda \underbrace{\sum_{\text{neighbors } i,j} \rho\left(D(i) - D(j)\right)}_{\textit{smoothness term}}$$

$$\boxed{l^* = arg \min_l E = \sum_p E_p(l_p; W) + \sum_p \sum_{q \in \mathcal{N}_p} E_{pq}(l_p, l_q)}$$

General Formulation - Recap

- Find the label l_p for each pixel p, for instance, by minimizing the following objective consisting of the data fidelity E_p and the prior term E_{pq}

$$l^* = arg\min_{l} \; E = \sum_{p} E_p(l_p; W) + \sum_{p} \sum_{q \in \mathcal{N}_p} E_{pq}(l_p, l_q)$$

Evaluating matching evidences with local image descriptors or matching similarity measures

Enforcing **the spatial smoothness constraint**

General Formulation: Local vs. Global?

- **Local approaches**
  - Using the data fidelity term only
  - Typically, aggregating the data cost with Edge-Aware Filtering (EAF)

$$E = \sum_p E_p(l_p; W)$$

Cost Volume Filtering, CVPR 2012
PatchMatch Filter, CVPR 2013

- **Global approaches**
  - Using both the data fidelity and prior terms
  - **Optionally**, aggregating the data cost with edge-aware filtering for stronger performance

$$E = \sum_p E_p(l_p; W) + \sum_p \sum_{q \in \mathcal{N}_p} E_{pq}(l_p, l_q)$$

$|W| = 1$, No cost aggregation

Belief Propagation, IJCV 2006
PatchMatch Belief Propagation, IJCV 2014
Sped-up PatchMatch Belief Propagation, ICCV 2015

Efficient Edge-Aware Filtering (EAF) as a fast alternative to global labeling optimization

[Petschnigg et al. SIGGRAPHY04] [Eisemann et al. SIGGRAPHY04]

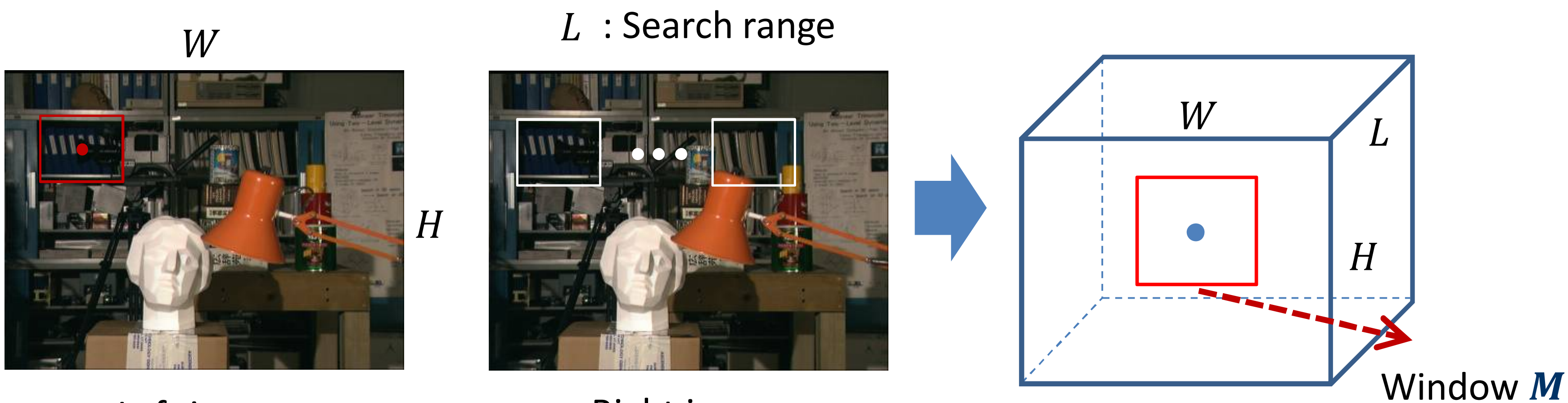
- Based on cross/joint (bilateral) filtering principles

PMF (PATCH-MATCH FILTER)

- J. Lu, H. Yang, D. Min*, and M. N. Do, 'PatchMatch Filter: Efficient Edge-Aware Filtering Meets Randomized Search for Fast Correspondence Field Estimation (CVPR), 2013. (oral presentation, acceptance rate < 4.0%, *: corresponding author)

Edge-Aware Filtering for Discrete Labeling Optimization

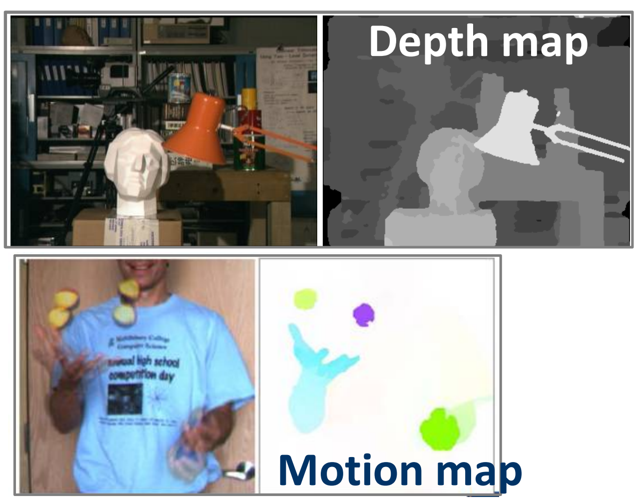
- **Labeling: assigning a label for all pixels (e.g. depth, motion)**



Left image

Right image

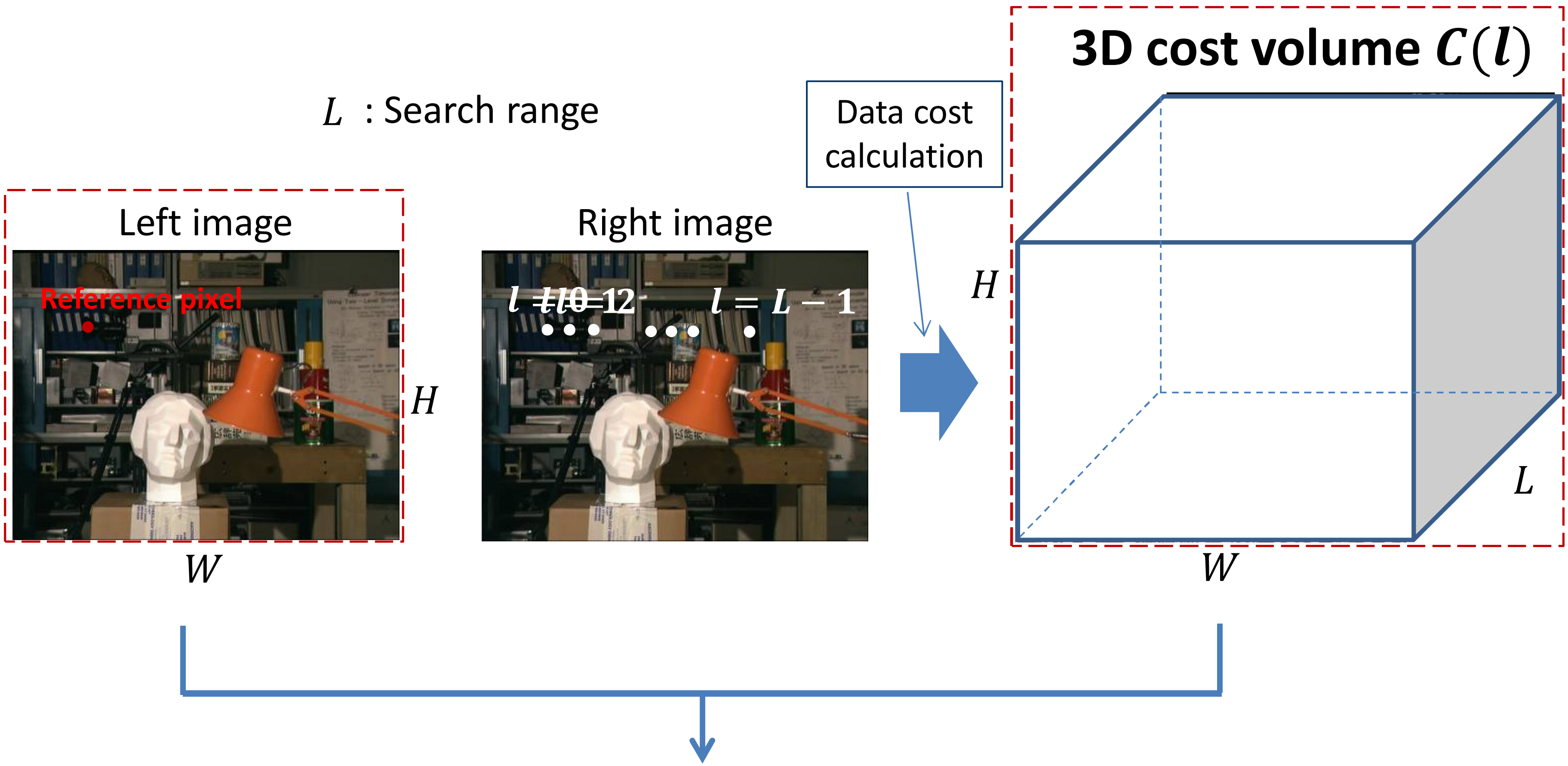
Examples of label maps



Applications using depth/motion

- View synthesis for 3DTV
- Frame up-conversion (30→ 60fps)
- 3D scene reconstruction
- Scene understanding
- 3D video editing

Edge-Aware Filtering for Discrete Labeling Optimization



JOINT Edge-Aware Filtering

for each slice of $C(l)$

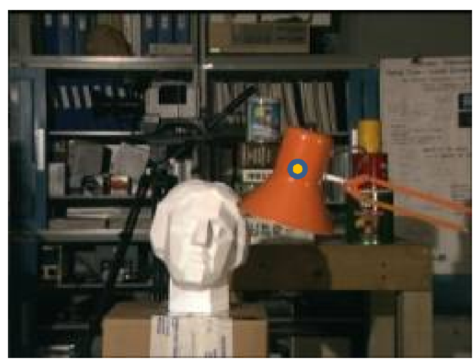
Edge-Aware Filtering for Discrete Labeling Optimization

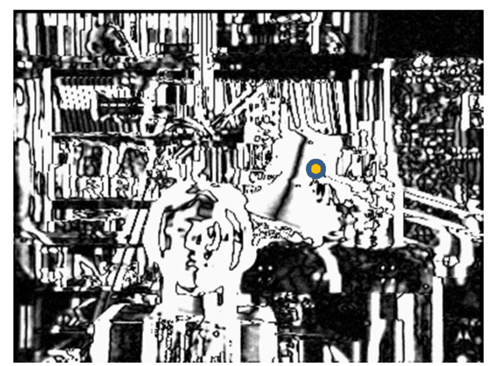
- Based on cross/joint (bilateral) filtering principles
- Cost volume filtering – repeated cross/joint filtering
- Runtime is often independent of the filter kernel size *m*

$$Output_{Cost} = \mathbf{EAF}(Color, Ini_{Cost})$$

$$\tilde{C}_p(l) = \sum_{q \in W_p(r)} \omega_{q,p}(I) C_q(l)$$

While a label
$l = 0 \rightarrow L - 1$





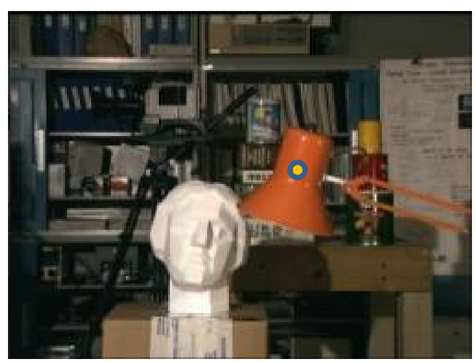
[Yoon & Kweon,PAMI06], [Rhemann et al.,CVPR11]

Edge-Aware Filtering for Discrete Labeling Optimization

Simple WTA

$$d(p) = arg\min_{l} \tilde{C}_p(l)$$

$$\tilde{C}_p(l) = \sum_{q \in W_p(r)} \omega_{q,p}(I) C_q(l)$$

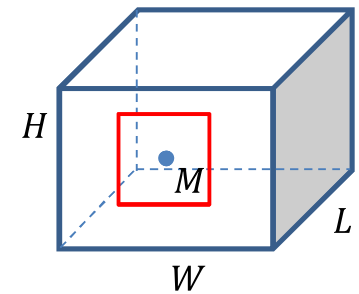


[Yoon & Kweon,PAMI06], [Rhemann et al.,CVPR11]

Fast Cost-Volume Filtering for Visual Correspondence and Beyond
(CVPR 2011, PAMI 2013)

- Reducing computational cost in terms of M
  - By using O(1) time edge-aware filtering (EAF): **Guided Filter (GF)**

$$O(IML) \dashrightarrow O(IL)$$



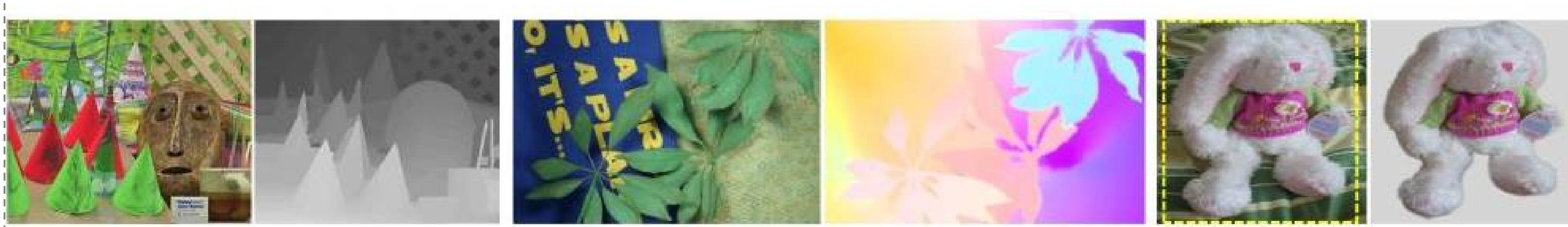
I: image size ($H \times W$), M: filter size, L: label size

Fast Cost-Volume Filtering for Visual Correspondence and Beyond

IEEE CVPR 2011

Christoph Rhemann[1] Asmaa Hosni[1] Michael Bleyer[1] Carsten Rother[2] Margrit Gelautz[1]

1 Vienna University of Technology, Austria
2 Microsoft Research Cambridge, UK

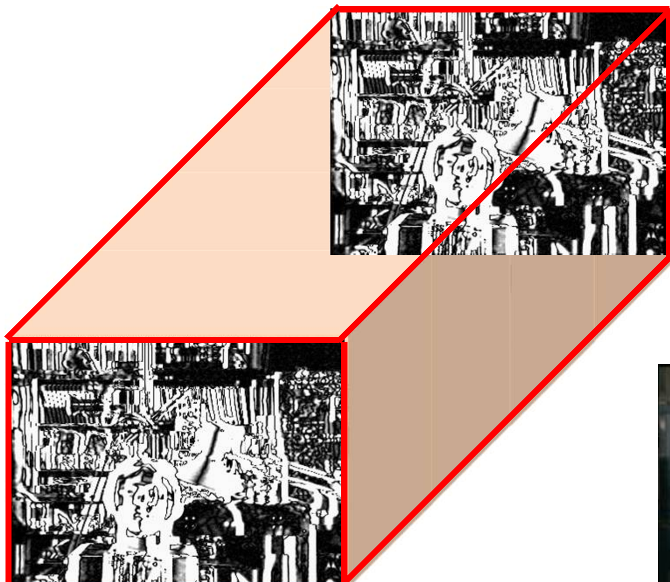


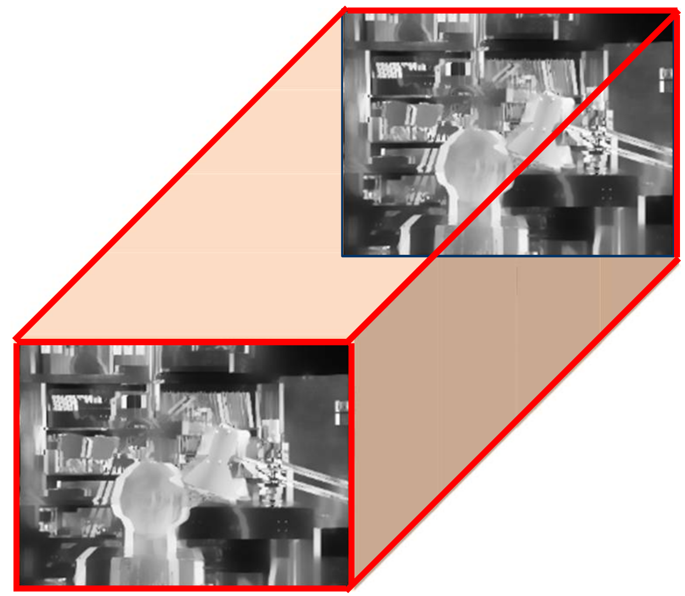
Stereo Optical Flow Interactive Segmentation

But, the curse of the label search space

$\mathcal{L} = \{0, 1, ..., L-1\}$

O(*I***L***) !!**





Also said for recent *filter-based mean-field inference for random fields* [Vineet et al. ECCV12]

The label space can be HUGE

- Two-dimensional motion search
- Displacement in subpixel accuracy
- Over-parameterized surface or motion modeling
- …

- e.g. motion search range in [-40, 40]*[-40, 40] * 8 * 8 → $\boldsymbol{L = 410,000}$ labels! → $\mathbf{410,000}$ joint filtering!

The label space can be HUGE

- Two-dimensional motion search
- Displacement in subpixel accuracy
- Over-parameterized surface or motion modeling
- …

- e.g. motion search range in [-40, 40]*[-40, 40] * 8 * 8 → $L = 410,000$ labels! → $410,000$ joint filtering!

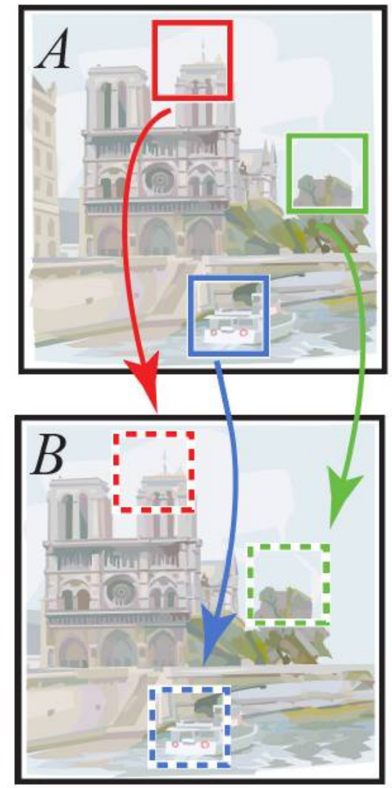
Too slow to stop at every floor

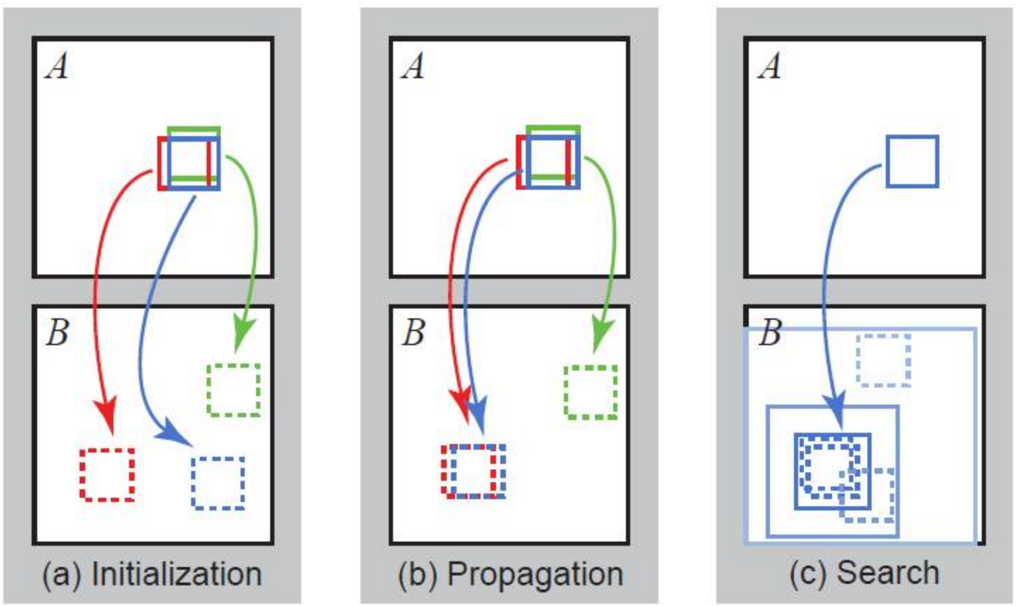
O(*I*L*) !!**

PatchMatch for Approximate Nearest-Neighbor Field (ANNF)

[Barnes et al., SIGGRAPH09, ECCV10]

- Find for every patch in A the nearest neighbor in B under a patch distance metric
- **Iterative *propagation & random search***





(a) Initialization (b) Propagation (c) Search

$O(I*M*\log L)$!!

I: image size ($H \times W$), M: filter size, L: label size

PatchMatch for Local Labeling Optimization

Toy example

A set of label candidates $L = \{0,1,\dots,99\}$

$E(p,l)$: Energy function to be minimized
at pixel p with label l

$D^t(p)$: Label map at t^{th} iteration of pixel p

1. Random Initialization of $D^0(p)$

1	10	37	80
59	20	75	95
72	41	28	50
55	30	92	62

PatchMatch for Local Labeling Optimization

Toy example

A set of label candidates $L = \{0, 1, \ldots, 99\}$

$E(p, l)$: Energy function to be minimized at pixel p with label l

$D^t(p)$: Label map at t^{th} iteration of pixel p

1. Random Initialization of $D^0(p)$

1	10	37	80
59	20	75	95
72	41	28	50
55	30	92	62

2. Propagation

15	20	35	73
51	30	75	95
72	41	28	50
55	30	92	62

$$D^t(p) = \underset{a \in \{35,30,75\}}{\arg\min} E(p, a)$$

PatchMatch for Local Labeling Optimization

Toy example

A set of label candidates $L = \{0,1,\ldots,99\}$

$E(p,l)$: Energy function to be minimized at pixel p with label l

$D^t(p)$: Label map at t^{th} iteration of pixel p

1. Random Initialization of $D^0(p)$

1	10	37	80
59	20	75	95
72	41	28	50
55	30	92	62

2. Propagation

15	20	35	73
51	30	30	95
72	41	28	50
55	30	92	62

$$D^t(p) = \underset{a \in \{35,30,75\}}{\arg\min}\; E(p,a)$$

PatchMatch for Local Labeling Optimization

Toy example

A set of label candidates $L = \{0,1,\ldots,99\}$

$E(p,l)$: Energy function to be minimized at pixel p with label l

$D^t(p)$: Label map at t^{th} iteration of pixel p

1. Random Initialization of $D^0(p)$

1	10	37	80
59	20	75	95
72	41	28	50
55	30	92	62

2. Propagation

15	20	35	73
51	30	30	95
72	41	28	50
55	30	92	62

$$D^t(p) = \underset{a \in \{35,30,75\}}{\arg\min}\, E(p,a)$$

15	20	35	73
51	30	60	95
72	41	28	50
55	30	92	62

3. Random Search

$$D^t(p) = \underset{a \in A}{\arg\min}\, E(p,a)$$

$$A = \left\{30 + \frac{R}{2^i} \middle| i = 1,\ldots,M\right\}$$

PatchMatch for Local Labeling Optimization

Toy example

A set of label candidates $L = \{0,1,\ldots,99\}$

$E(p,l)$: Energy function to be minimized at pixel p with label l

$D^t(p)$: Label map at t^{th} iteration of pixel p

1. Random Initialization of $D^0(p)$

1	10	37	80
59	20	75	95
72	41	28	50
55	30	92	62

2. Propagation

15	20	35	73
51	30	30	95
72	41	28	50
55	30	92	62

$$D^t(p) = \underset{a \in \{35,30,75\}}{\arg\min}\; E(p,a)$$

Repeat $\boldsymbol{T}$ times!

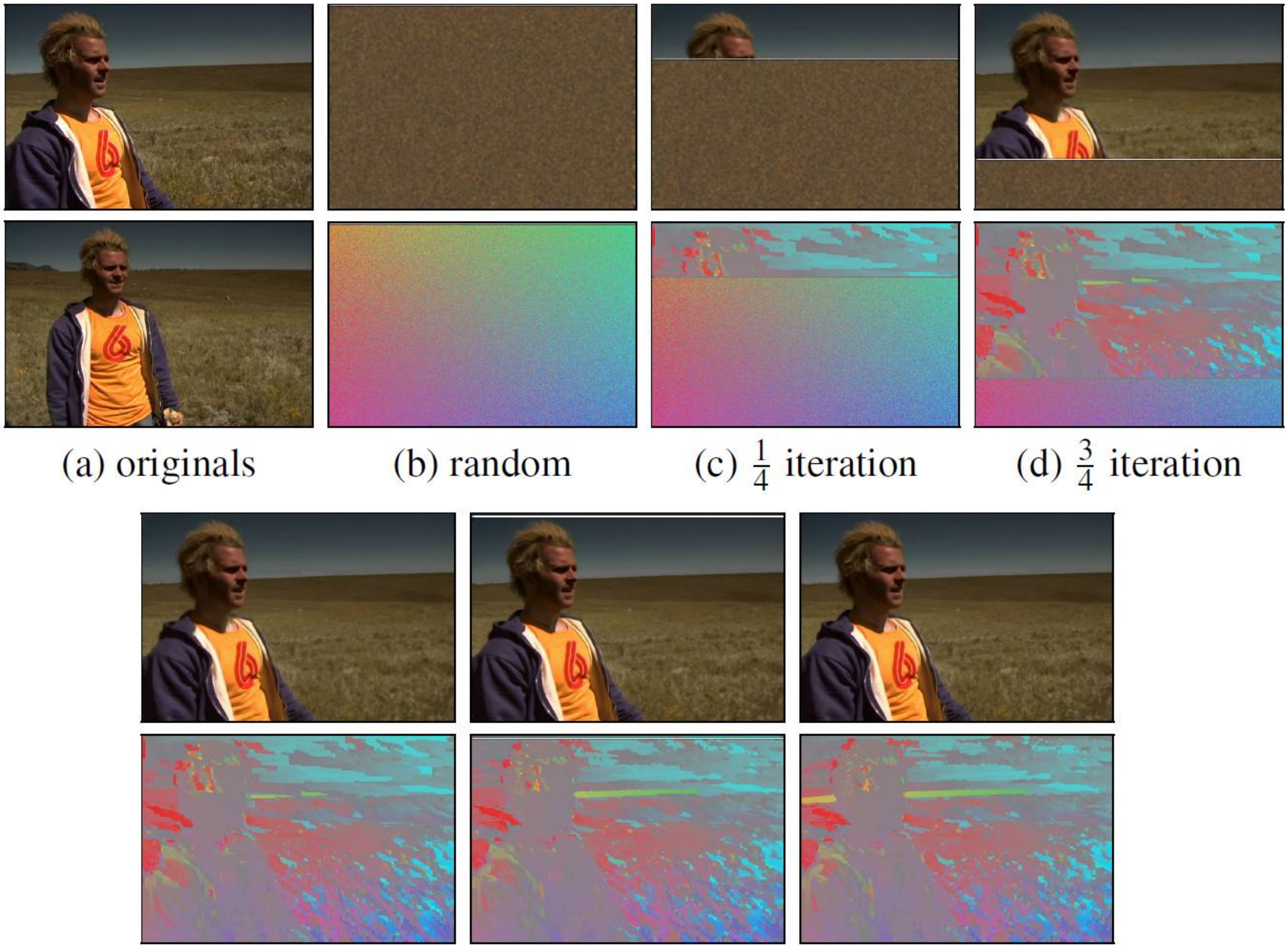
15	20	35	73
51	30	60	95
72	41	28	50
55	30	92	62

3. Random Search

$$D^t(p) = \underset{a \in A}{\arg\min}\, E(p,a)$$

$$A = \left\{30 + \frac{R}{2^i} \middle| i = 1, \ldots, M\right\}$$

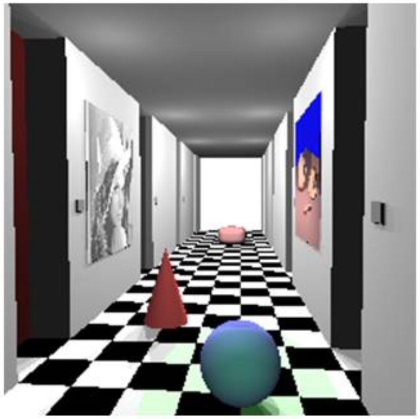
PatchMatch for Local Labeling Optimization



(a) originals (b) random (c) $\frac{1}{4}$ iteration (d) $\frac{3}{4}$ iteration

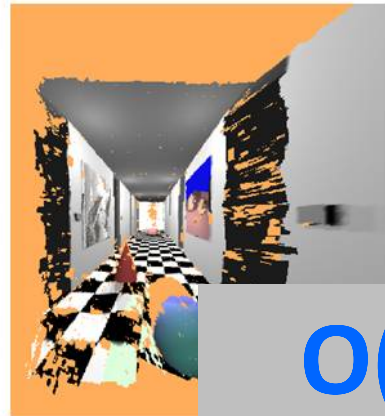
(e) 1 iteration (f) 2 iterations (g) 5 iterations

Related work dealing with the huge label space



Left image

Disparity map

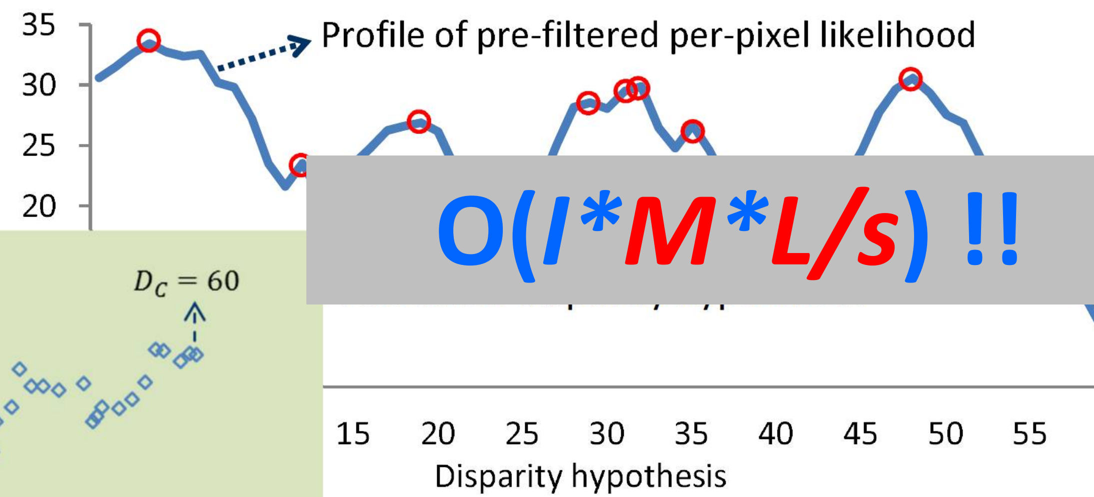


3D reconstruction

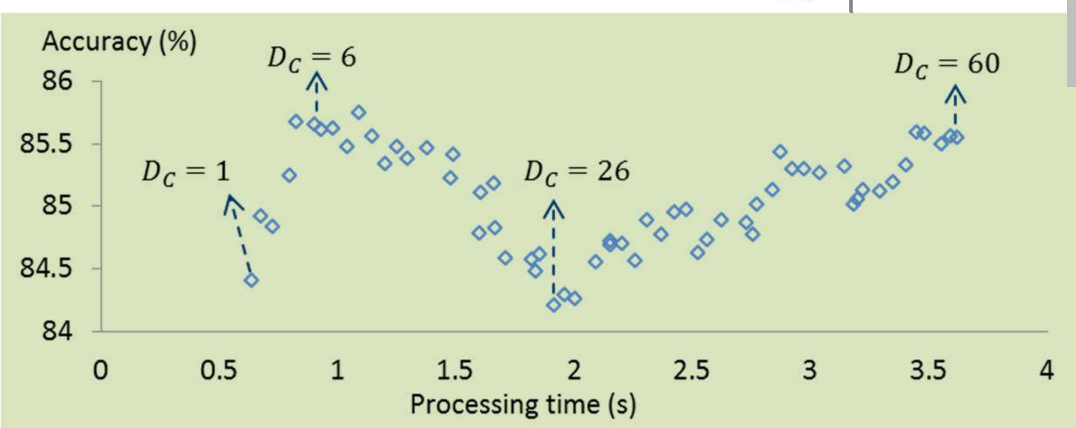
PatchMatch stereo
[Bleyer et al., BMVC11]

O(*I*M*logL*) !!

Histogram-based prefiltering [Min et al., ICCV11]



O(*I*M*L/s*) !!

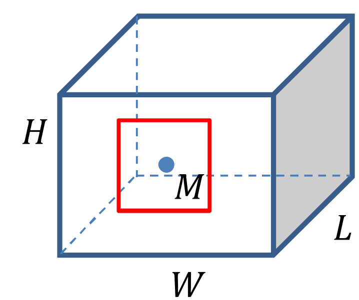


I: image size ($H \times W$), ***M***: filter size, ***L***: label size

But, the cost is still huge: New approach?

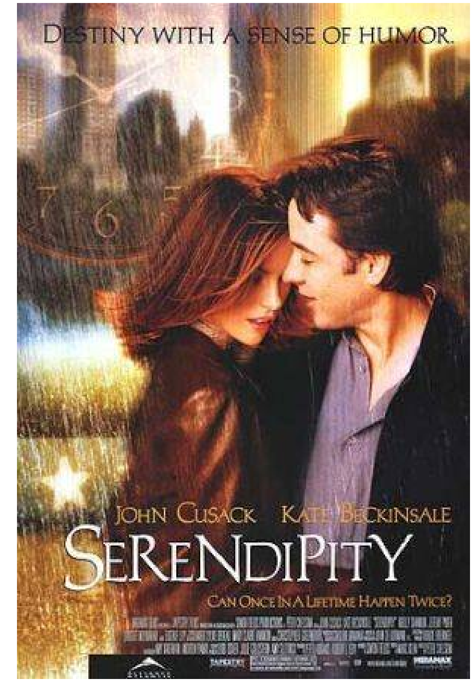
- Computational complexity of local labeling optimization
  - Brute force approach: $\boldsymbol{O(IML)}$
  - CostFilter (CVPR 2011, PAMI 2013): $\boldsymbol{O(IL)}$
  - PatchMatch (SIGGRAPH 2009, ECCV 2010): $\boldsymbol{O(IMlogL)}$
  - Histogram-based prefiltering (ICCV 2011): $\boldsymbol{O(IML/10)}$

$\boldsymbol{I}$: image size ($H \times W$), $\boldsymbol{M}$: filter size,
$\boldsymbol{L}$: label size

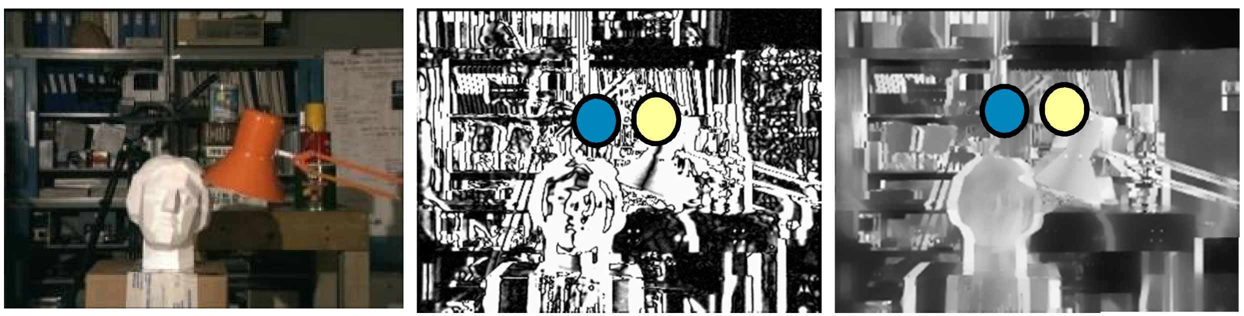



PatchMatch Filter
O(I*log$L$)
EAF
O(I*L)
PatchMatch
O(I*M*log$L$)
Our goal is to find a bridge to
enjoy high “throughput” !!
Photo courtesy: www.pamitc.org/cvpr13/attending.php

Meeting the two is never easy



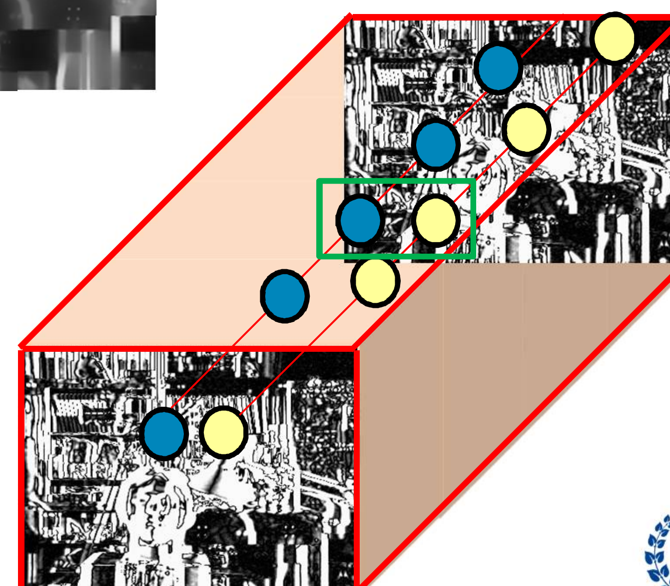
- Significantly different objectives
- Disparate computation pattern
- Disparate memory access pattern



EAF: Highly regular and deterministic computing

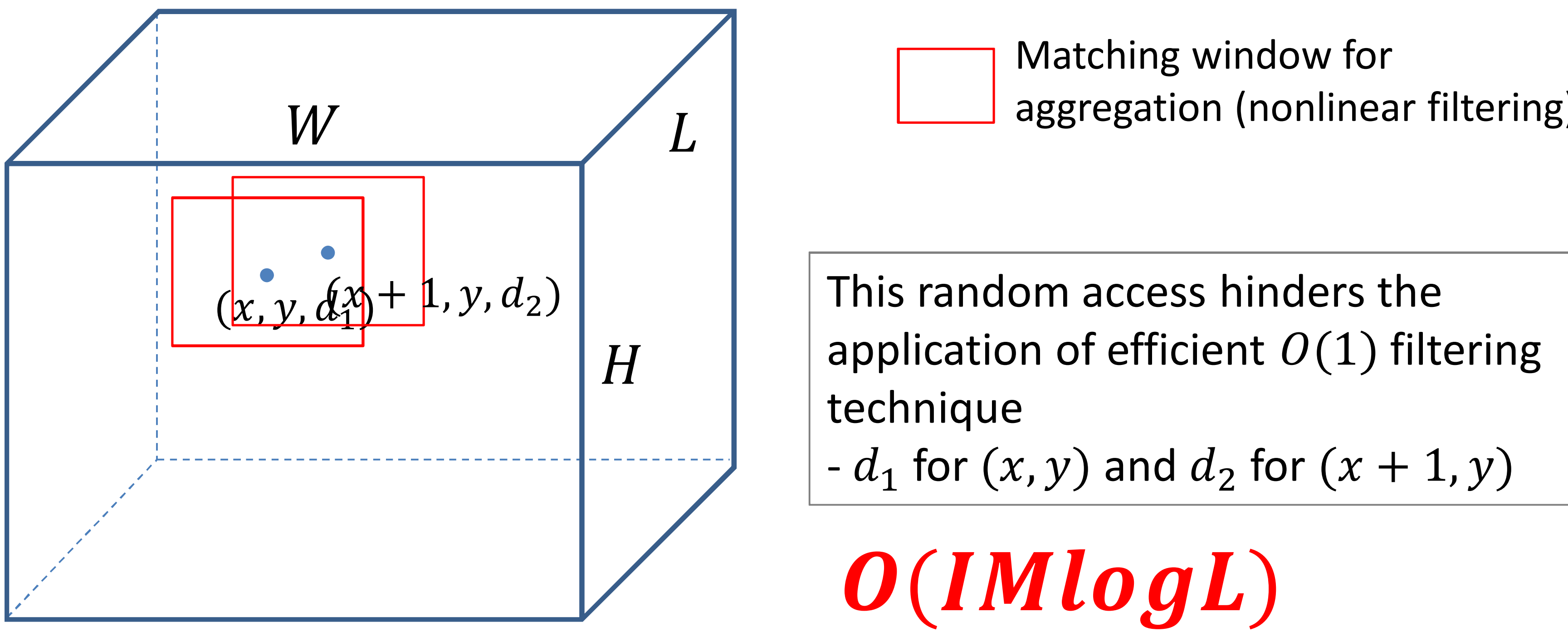
Vs

PM: Random and fragmented data access



Random Access on Label Space Makes Problem **DIFFICULT**

- Pixel-wise randomized search of original PatchMatch
  - Fragmental data access on 3D cost volume



3D cost volume

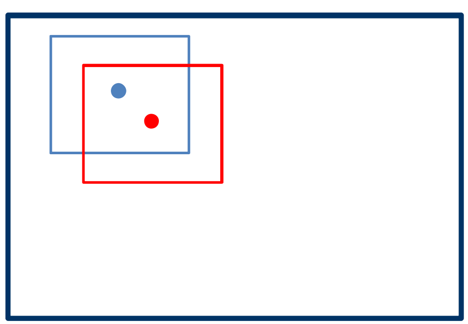
This random access hinders the application of efficient $O(1)$ filtering technique
- d_1 for (x, y) and d_2 for $(x+1, y)$

$$O(IMlogL)$$

I: image size $(H \times W)$, M: filter size, L: label size

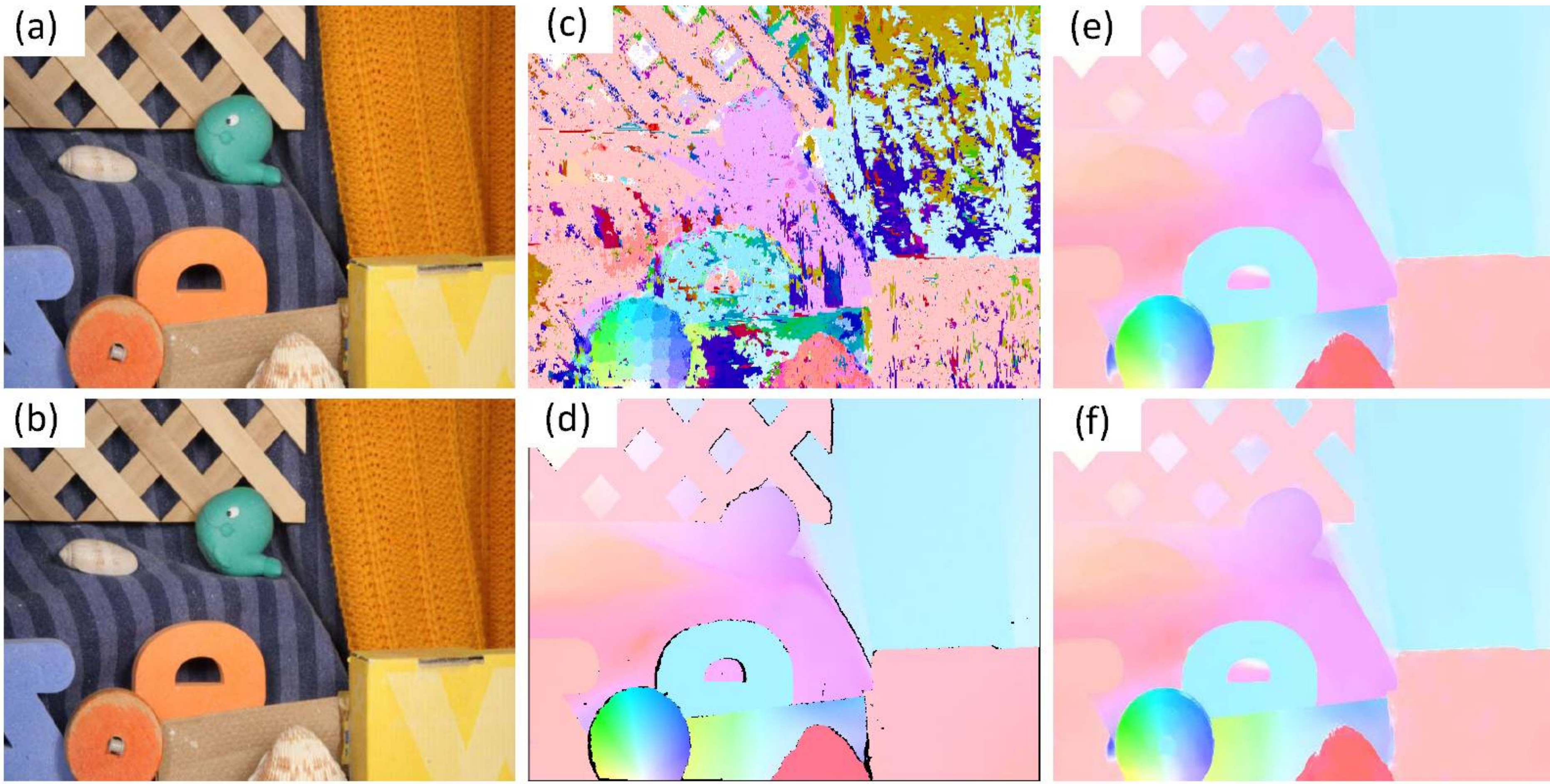
$O(1)$ filtering needs redundancy!

- Redundancy of simultaneously computing a weighted sum for ***all*** pixels
  - ***Guided filter*** (ECCV 2010, PAMI 2013): Multiple number of integral sum (box filtering)
  - ***Recursive filter of Domain Transform*** method (SIGGRAPH 2011): Recursive propagation of aggregated data in causal and non-causal manners
  - $\boldsymbol{O(1)}$ ***Bilateral Filter on bilateral grid*** (ECCV 2006): Linear Gaussian filtering on high dimensional volume



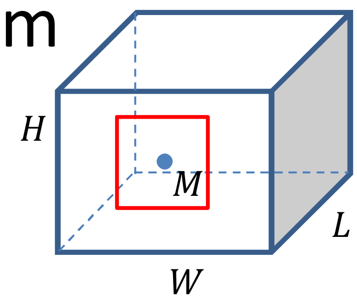
The filtered data of
should be reused for filtering

We did it: Ours (f) runs 10x faster than CostFilter (e), with even higher accuracy



PatchMatch Filter (PMF)

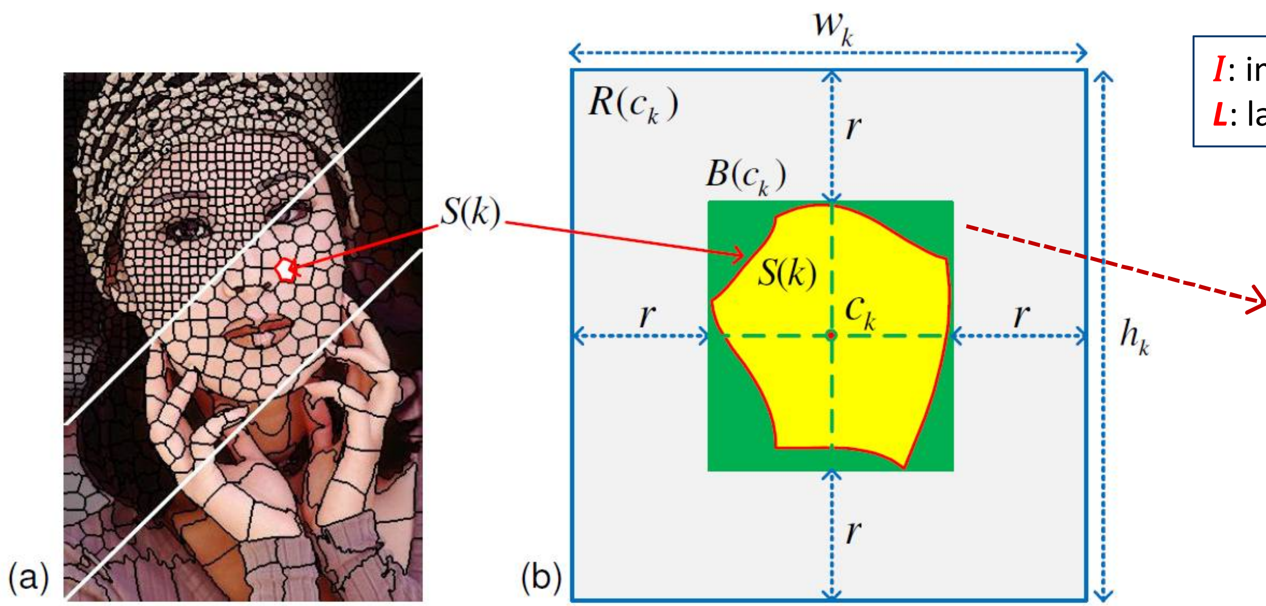
- *Super-pixel* based randomized search algorithm
  - Collaborative filtering within a single super-pixel



- Efficient filtering + PatchMatch algorithm
  - Collaborative randomized search

$$O(IL) \quad + \quad O(IMlogL) \quad \dashrightarrow \quad O(IlogL)$$

I: image size ($H \times W$), M: filter size, L: label size

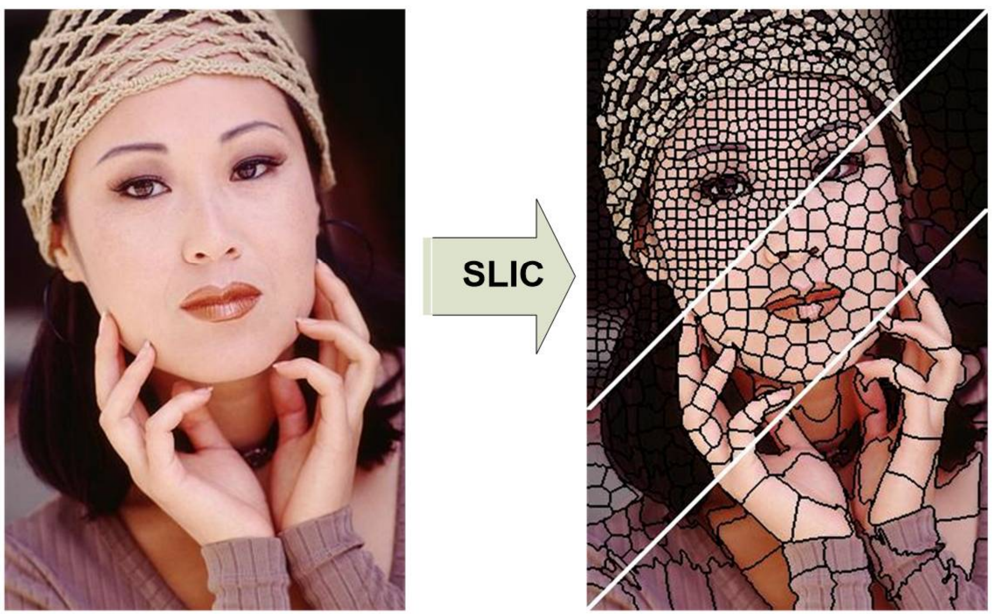


Processing unit for filtering

Segments as the bridge

• Labeling solutions are spatially smooth and discontinuities-aligned
➔ **Collaborative label search and propagation**
➔ Extends the propagation range

• **The efficiency of EAF** comes from high computational redundancy for **shared computation reuse**



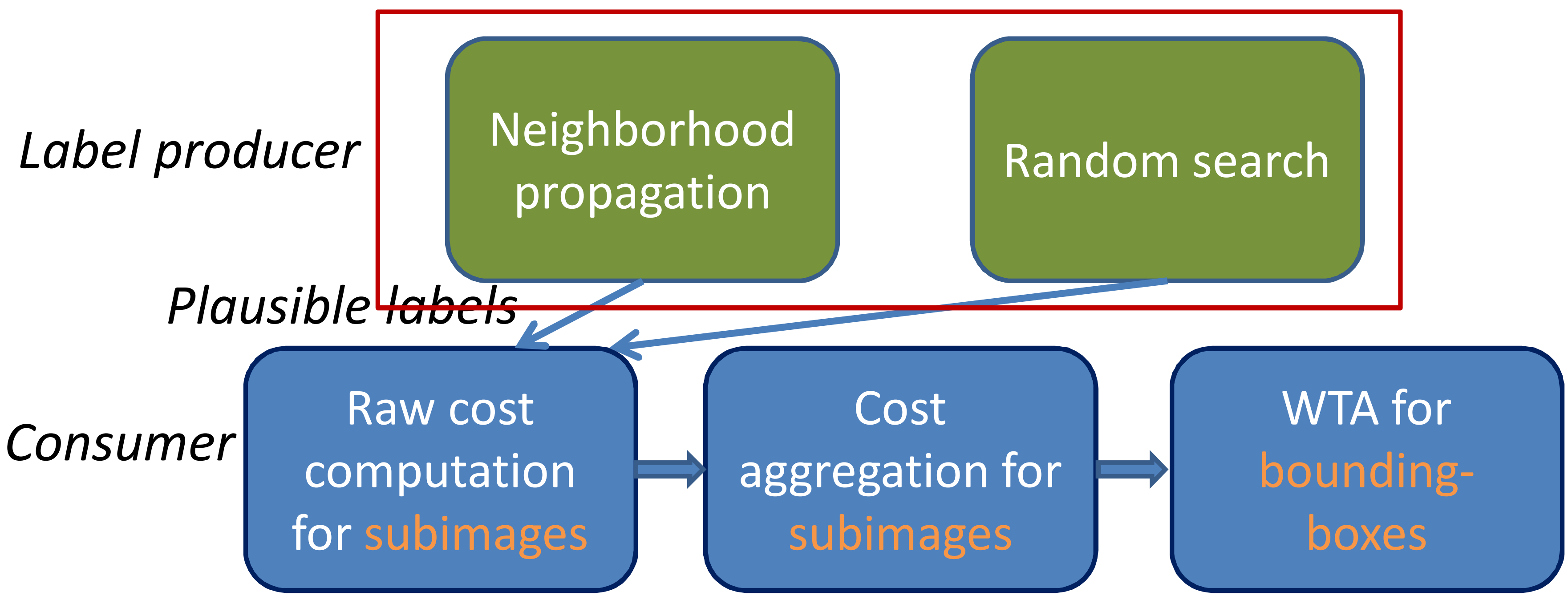
Note)
A simple method, dividing an image into **non-overlapped rectangular blocks**, is also possible!
→ But, this makes the algorithm being converged **much slower**!

[Achanta et al. PAMI12]

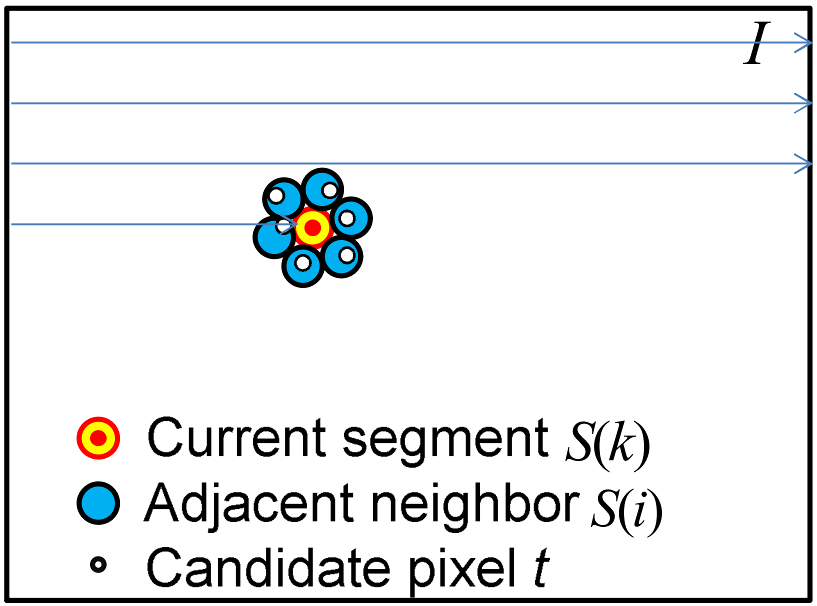
Baseline PMF algorithm: General recipe

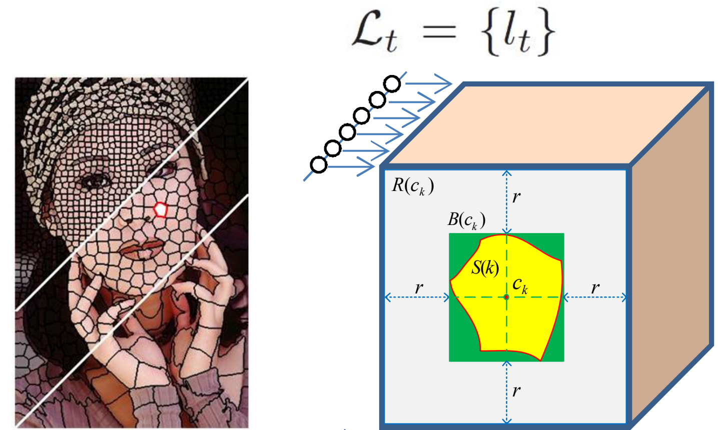
1. Initial label assignment to each segment
2. **Process each segment in scan order iteratively**
   - For the current segment, evaluate the candidate labels generated from two sources: *propagation* & *random search*

Note that the **cost aggregation (filtering)** is done for **each segment**
the **label decision (WTA)** is done for **each pixel**

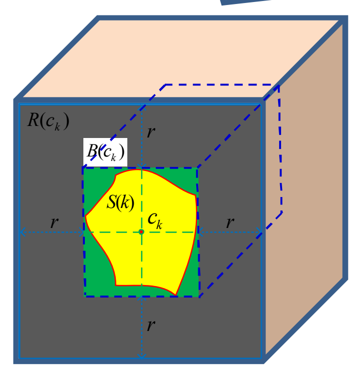


Neighborhood label propagation & evaluation



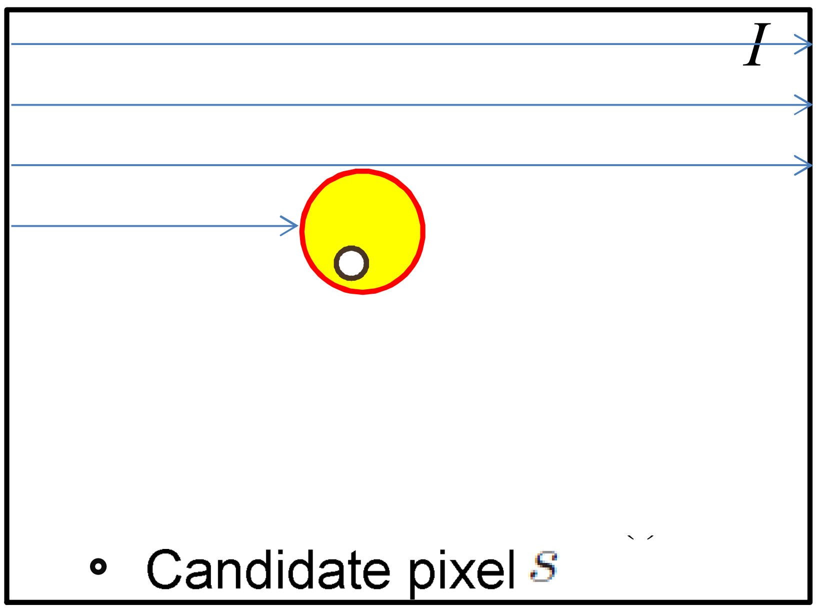


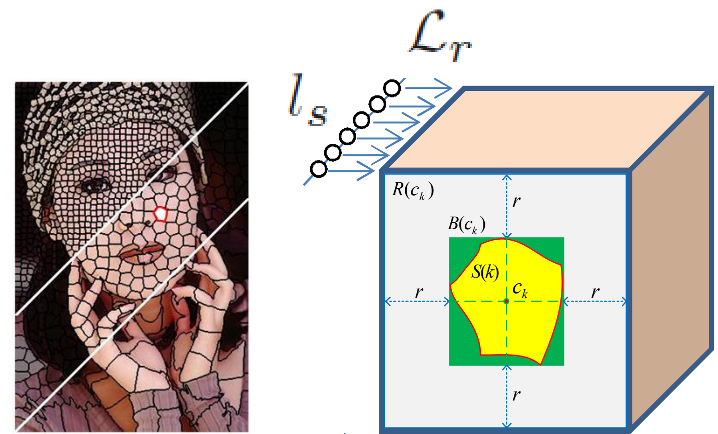
$$\mathbf{f} : \mathbf{C}\left(R(c_k), \{l \in \mathcal{L}_t\}\right) \mapsto \tilde{\mathbf{C}}\left(B(c_k), \{l \in \mathcal{L}_t\}\right)$$



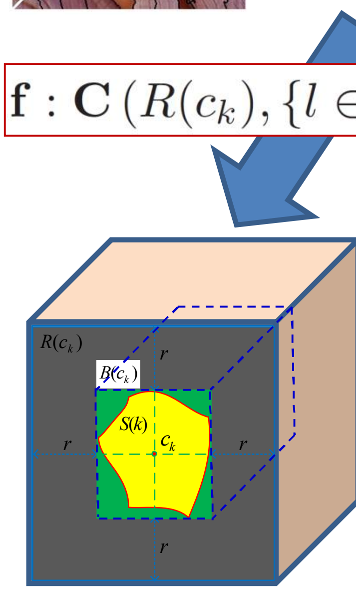
Random search & evaluation

Note that the filtering is done for each segment,
but the label decision is done for each pixel





$$\mathbf{f} : \mathbf{C}\left(R(c_k), \{l \in \mathcal{L}_r\}\right) \mapsto \tilde{\mathbf{C}}\left(B(c_k), \{l \in \mathcal{L}_r\}\right)$$

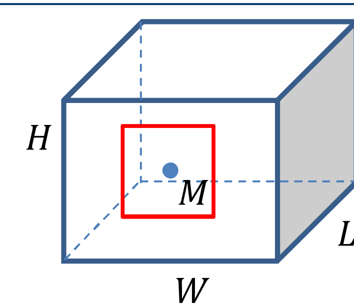


A list records the labels visited for each segment *S*(*k*), so NO subimage filtering for any revisited label.

Complexity Comparison

- Computational complexity of local labeling optimization
  - Brute force approach: $\boldsymbol{O(IML)}$
  - CostFilter (CVPR 2011, PAMI 2013): $\boldsymbol{O(IL)}$
  - PatchMatch (SIGGRAPH 2009, ECCV 2010): $\boldsymbol{O(IMlogL)}$
  - Histogram-based prefiltering (ICCV 2011): $\boldsymbol{O(IML/10)}$
  - **PatchMatch Filter (ours): $\boldsymbol{O(IlogL)}$**

$\boldsymbol{I}$: image size $(H \times W)$, $\boldsymbol{M}$: filter size, $\boldsymbol{L}$: label size



PMF for Stereo – Slanted surface handling

- **Label**: for each pixel *p*, find a 3D plane

$$\mathbf{l}_p = (a_p, b_p, c_p)$$

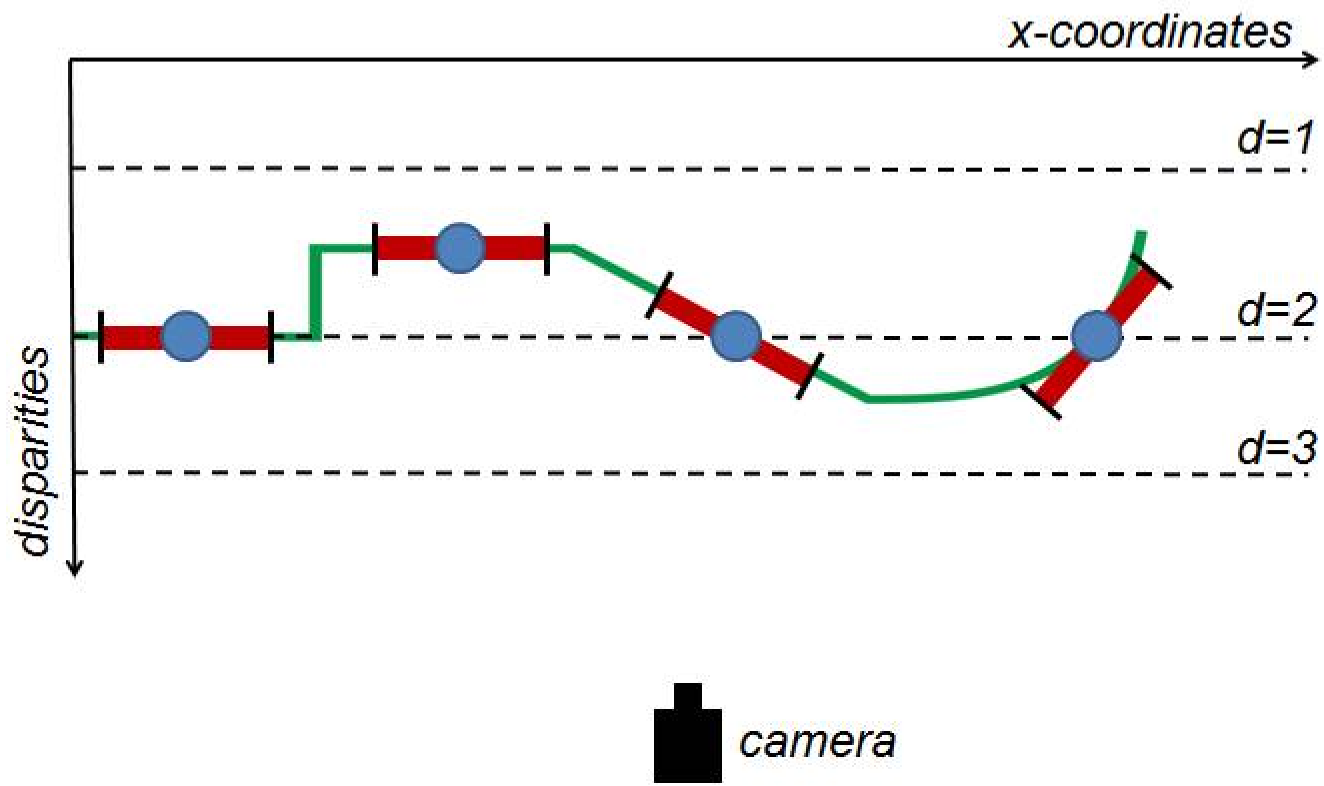


Image courtesy of [Bleyer et al., BMVC11]

PMF for Stereo – Slanted surface handling

- **Label**: for each pixel *p*, find a 3D plane $\mathbf{l}_p = (a_p, b_p, c_p)$
- Hypothetical correspondence location (*q*, *q'*)

$$x_{q'} = x_q - d_q = x_q - \mathbf{l}_p \cdot (x_q, y_q, 1)^\top \text{ , and } y_{q'} = y_q$$

- **Raw matching cost**

$$C_q(l) = (1-\beta) \cdot \min\left(\left\|I_q - I'_{q'}\right\|, \gamma_1\right) + \beta \cdot \min\left(\left\|\nabla I_q - \nabla I'_{q'}\right\|, \gamma_2\right)$$

- **PMF-based cost aggregation**
- **Post processing**
  - Cross-checking, plane extrapolation for unreliable pixels, weighted median filter

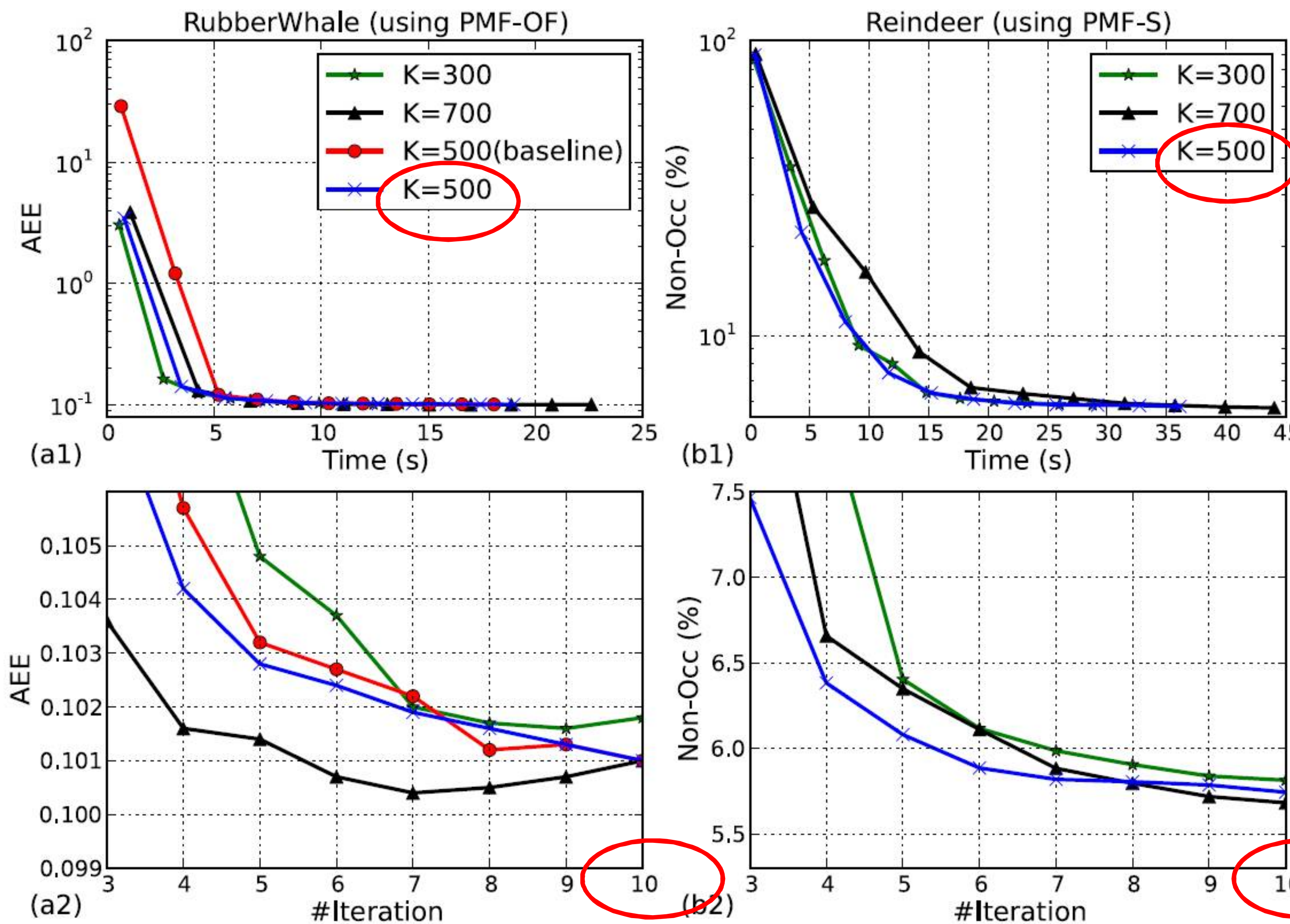
PMF for Optical flow

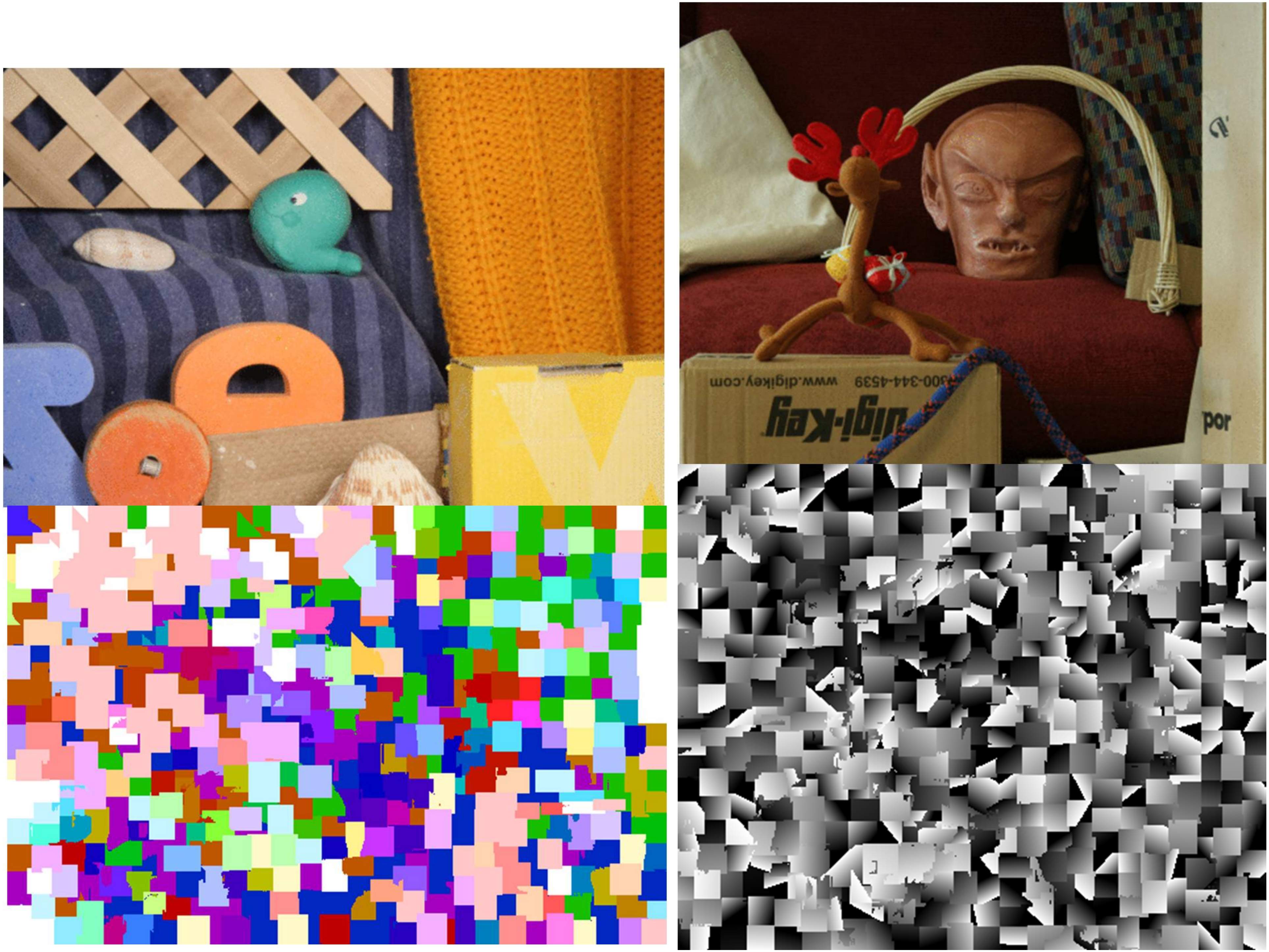
- **Label**: for each pixel *p*, find a disp. vector $\mathbf{l}_p = (u, v)$
- For sub-pixel accurate flow, upscaling (*u*,*v*)-dim. by 8
- Hypothetical correspondence location (*q*, *q'*=*q*+(*u*,*v*))
- **Raw matching cost**

$$C_q(l) = (1-\beta) \cdot \min\left(\left\|I_q - I'_{q'}\right\|, \gamma_1\right) + \beta \cdot \min\left(\left\|\nabla I_q - \nabla I'_{q'}\right\|, \gamma_2\right)$$

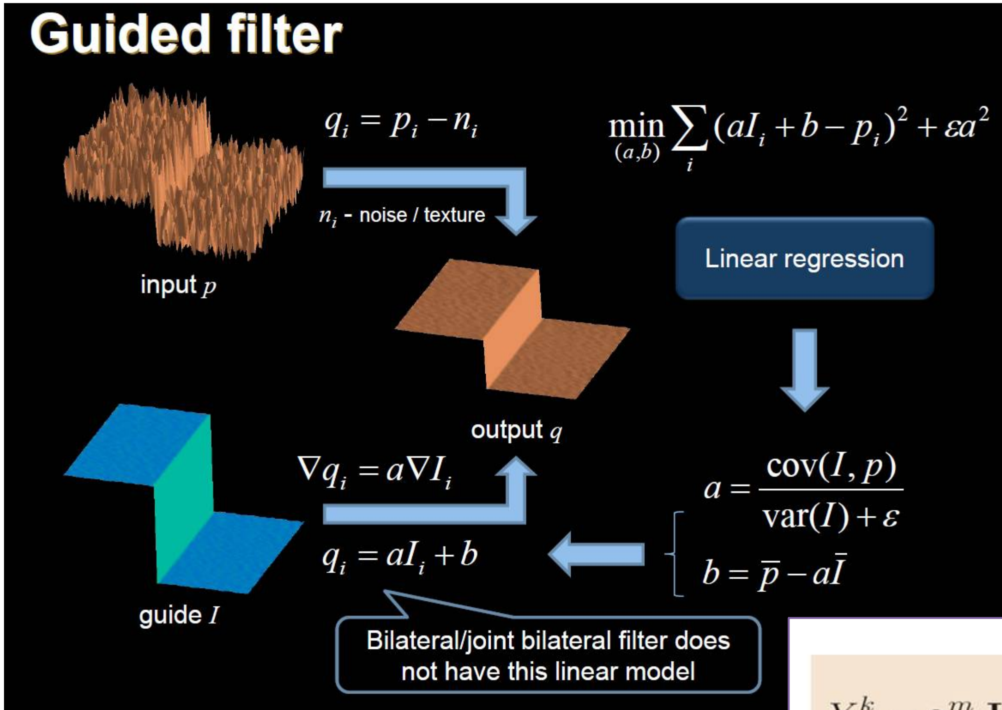
- **PMF-based cost aggregation**
- **Post processing**
  - Cross-checking, iterative weighted median filter, smoothing

Convergence and time-accuracy trade-off




www.digikey.com
digi-key

Guided Filter (GF) and Cross-based Local Multipoint Filtering (CLMF)

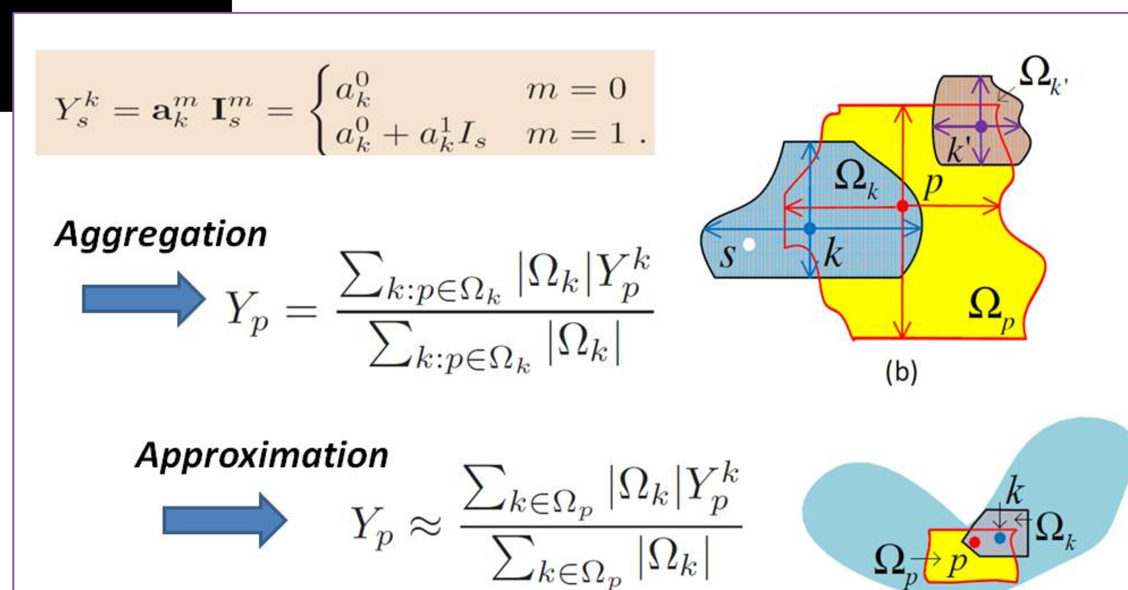


* Both O(1)-time algorithms with leading EAF performance

* *CLMF-0* is **2-3x** faster than *GF*, *CLMF-1* gives better quality

[Lu et al. CVPR12]

[He et al. ECCV10]



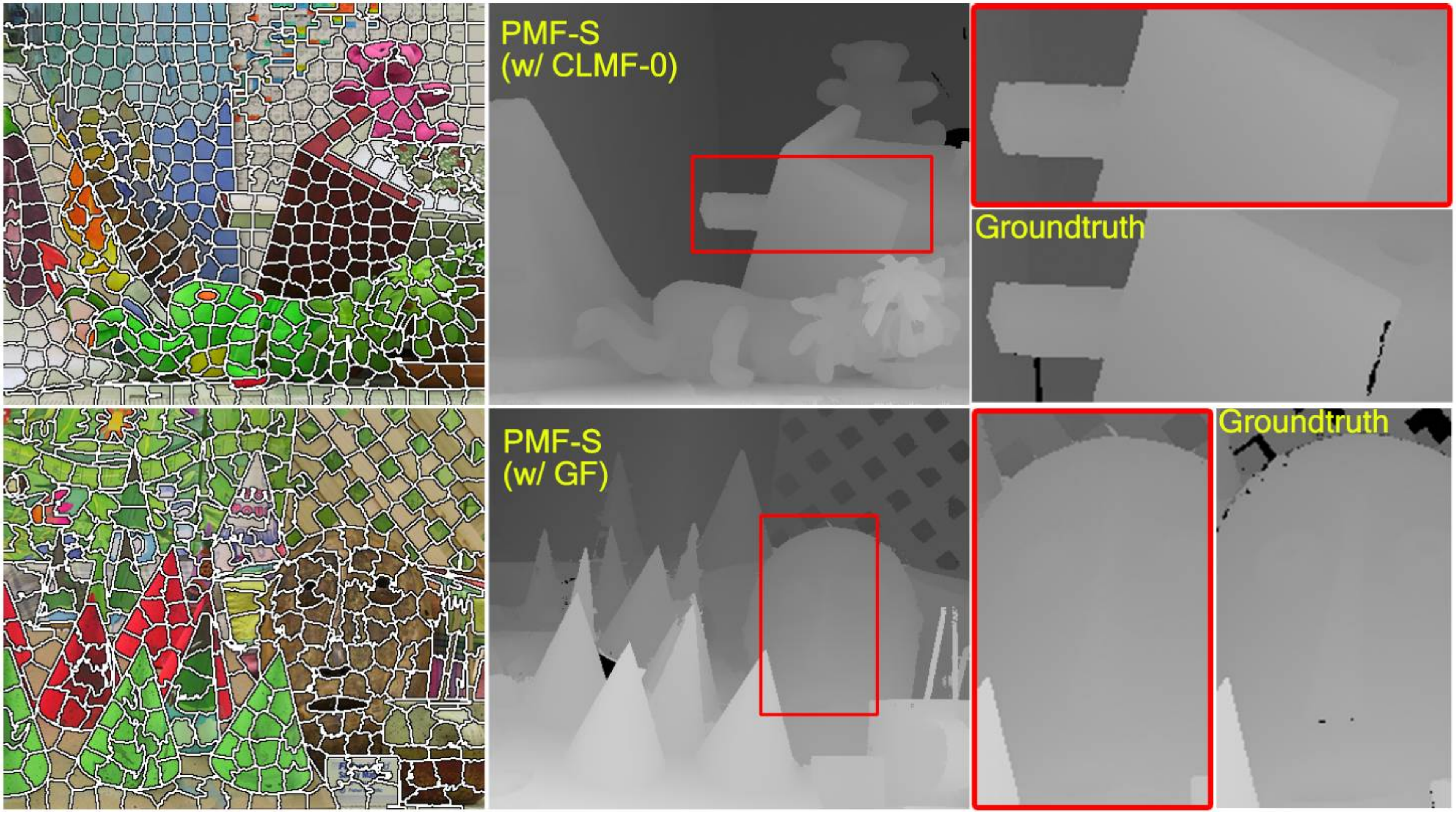
Middlebury stereo benchmark evaluation

Algorithm	Err. thre. = 1.0		Err. thre. = 0.5	
	Rank	Err. %	Rank	Err. %
PMF-S (w/ CLMF-0)	**15**	**4.04**	6	8.67
PMF-S (w/ GF)	16	4.06	**2**	**7.69**
PatchMatch [7]	18	4.59	8	9.91
PMBP [6]	21	4.46	4	8.77
PMF-C (w/ CLMF-0)	23	5.26	-	-
CostFilter (w/ GF) [17]	24	5.55	-	-
PMF-C (w/ GF)	25	5.48	-	-

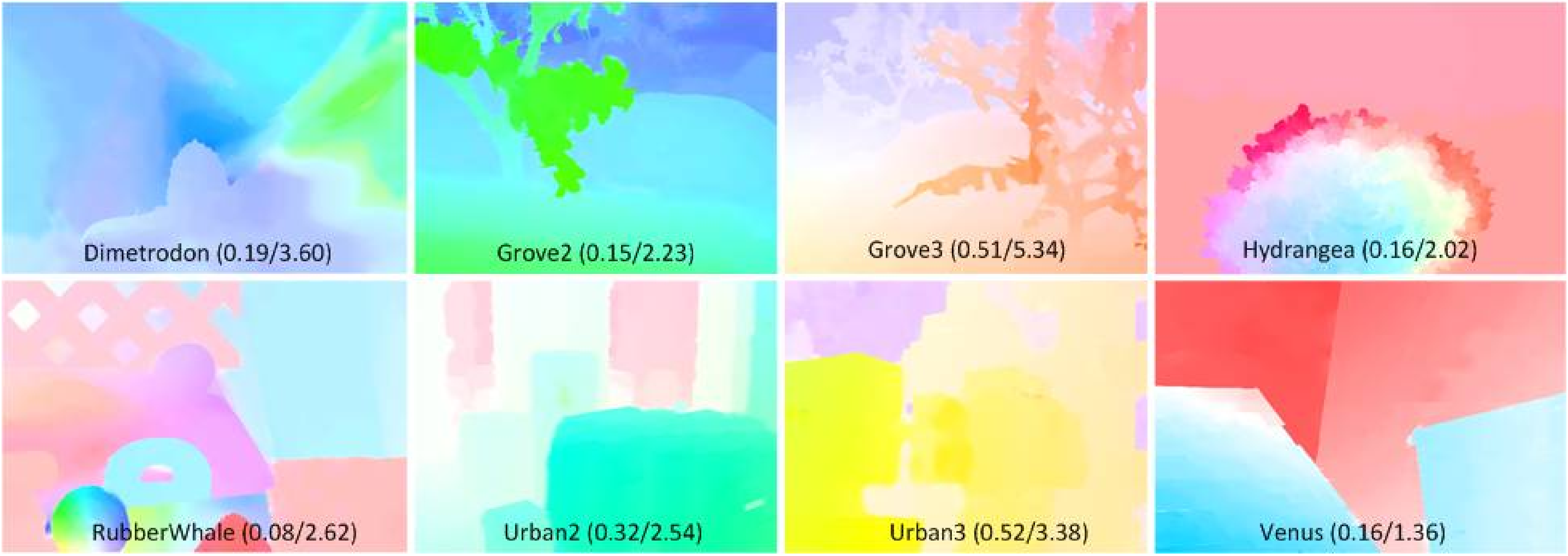
- Improvement of PMF due to implicit regularization by segment-based propagation
- **PMF does not sacrifice the matching accuracy (compared to the original PatchMatch), for improving the runtime efficiency!**

Algorithm	*Teddy*			*Cones*		
	nocc	all	disc	nocc	all	disc
PMF-S-GF	4.45_2	9.44_2	13.7_2	$\mathbf{2.89_1}$	8.31_2	$\mathbf{8.22_1}$
PMBP [6]	5.60_3	12.0_6	15.5_3	3.48_3	8.88_4	9.41_4
PMF-S-CLMF0	$\mathbf{4.07_1}$	10.5_3	$\mathbf{12.1_1}$	2.96_2	8.84_3	8.38_2
PatchMatch [7]	5.66_4	11.8_5	16.5_4	3.80_5	10.2_6	10.2_5

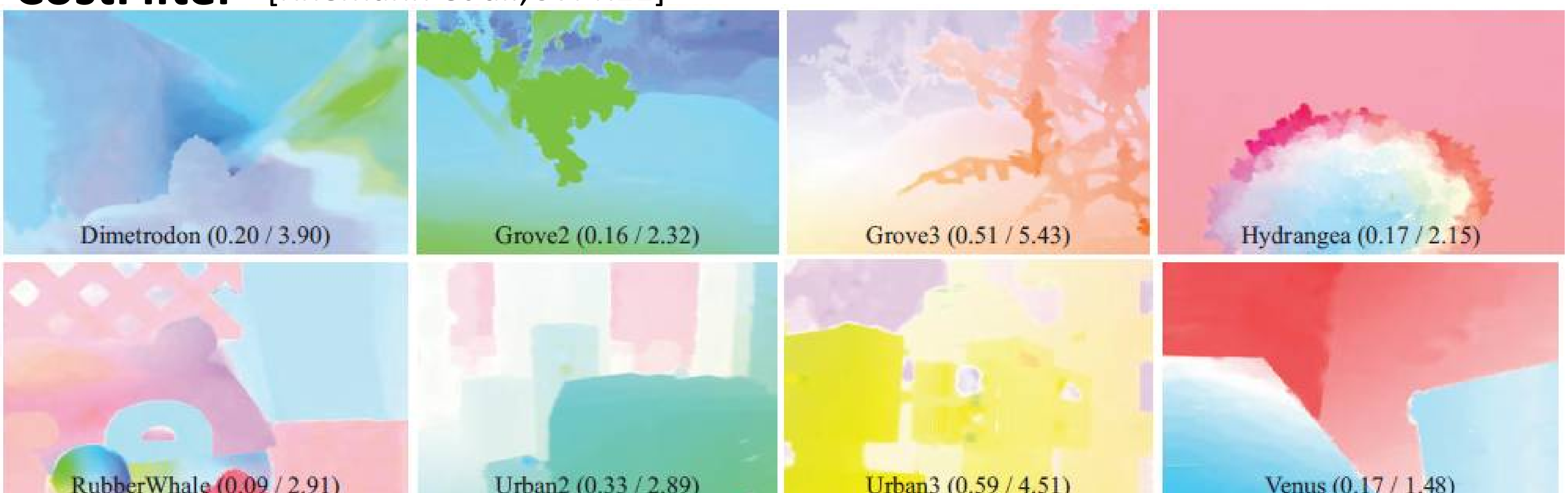
- PMF-C runs over 3-7x faster for high-res. stereo images (e.g. 1M pixels) than CostFilter
- PMF-S over 10x faster than PatchMatch Stereo[7]
- In any case, w/ CLMF-0 runs 2-3x than w/ GF


PMF-S
(w/ CLMF-0)
Groundtruth
PMF-S
(w/ GF)
Groundtruth

Our PMF



CostFilter [Rhemann et al.,CVPR11]



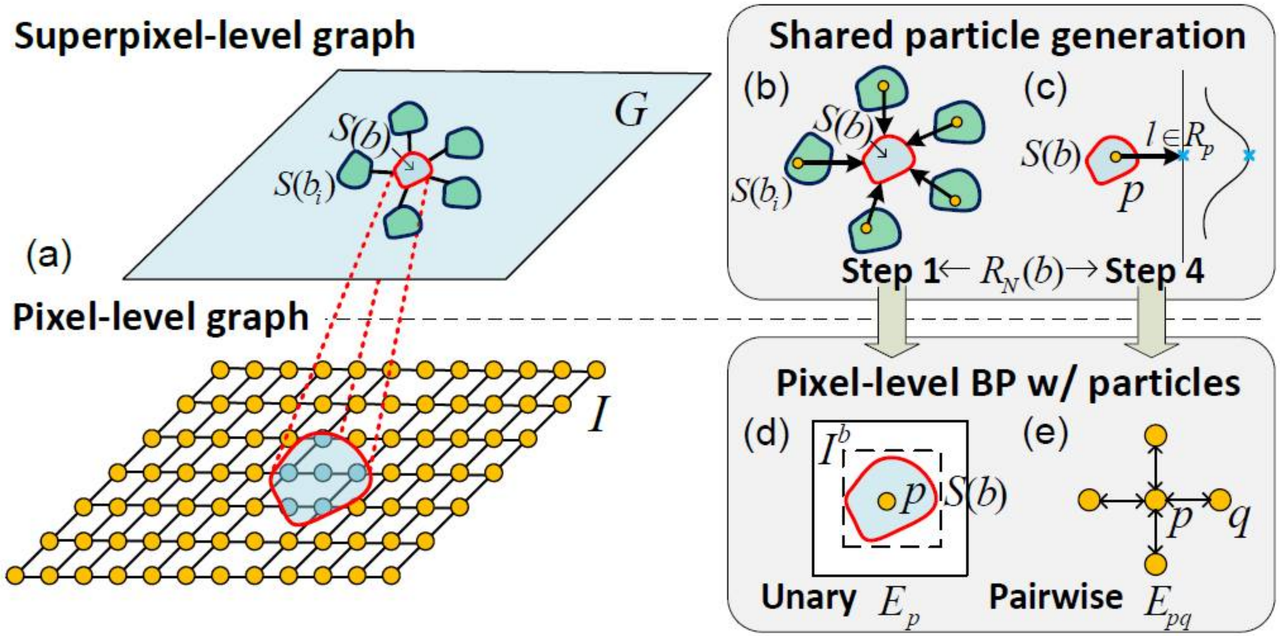
Middlebury optical flow evaluation

Algorithm	μRank	*Schefflera*	*Grove*	*Teddy*	sec
MDP-Flow2 [20]	5.0	(2,2,**1**)	(9,10,10)	(2,2,2)	342
PMF-GF	19.9	(5,5,8)	(4,4,3)	(3,**1**,7)	35
MDP-Flow	21.6	(6,8,28)	(21,21,26)	(44,47,43)	188
PMF-CLMF-0	22.5	(15,17,8)	(8,8,2)	(4,2,9)	18
CostFilter [17]	25.0	(4,4,13)	(6,7,4)	(9,18,9)	55*
DPOF [12]	31.2	(6,6,28)	(12,15,8)	(22,18,4)	287

Middlebury optical flow evaluation

Algorithm	μRank	*Schefflera*	*Grove*	*Teddy*	sec
MDP-Flow2 [20]	5.0	(2,2,**1**)	(9,10,10)	(2,2,2)	342
PMF-GF	19.9	(5,5,8)	(4,4,3)	(3,**1**,7)	35
MDP-Flow	21.6	(6,8,28)	(21,21,26)	(44,47,43)	188
PMF-CLMF-0	22.5	(15,17,8)	(8,8,2)	(4,2,9)	18
CostFilter [17]	25.0	(4,4,13)	(6,7,4)	(9,18,9)	55*
DPOF [12]	31.2	(6,6,28)	(12,15,8)	(22,18,4)	287

- PMF gives an order of magnitude runtime speedup
- PMF runs even over 30x faster than CostFilter[17] on the same PC



SPM-BP (SPED-UP PATCHMATCH BELIEF PROPAGATION)

- Y. Li, D. Min*, M. S. Brown, M. N. Do, and J. Lu, 'SPM-BP: Sped-up PatchMatch Belief Propagation for Continuous MRFs,' IEEE Int. Conf. on Computer Vision (ICCV), Dec. 2015. (oral presentation, acceptance rate < 4.0%, *: corresponding author)

Discrete Pixel-Labeling Optimization on MRF

- Many computer vision tasks can be formulated as a pixel-labeling problem on Markov Random Field (MRF)

Segmentation
$l = \{B, G\}$

Denoising
$l = intensity$

Stereo
$l = d$

Optical flow
$l = (u, v)$

$$E = \sum_p E_p(l_p; W) + \sum_p \sum_{q \in \mathcal{N}_p} E_{pq}(l_p, l_q)$$

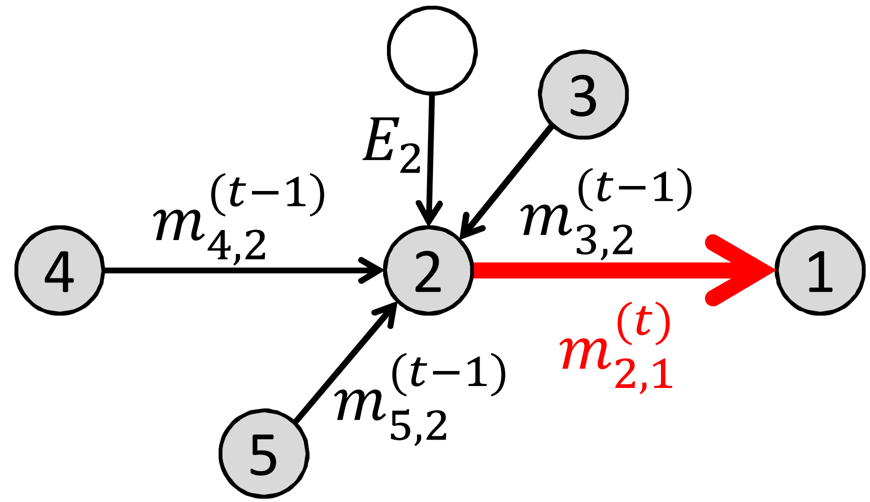
p: pixel, N_p: 4 neighbors

- Simple: data term + smoothness term
- Effective: labeling coherence, discontinuity handling
- Optimization: Graph Cut, $\boxed{\text{Belief Propagation}}$, etc

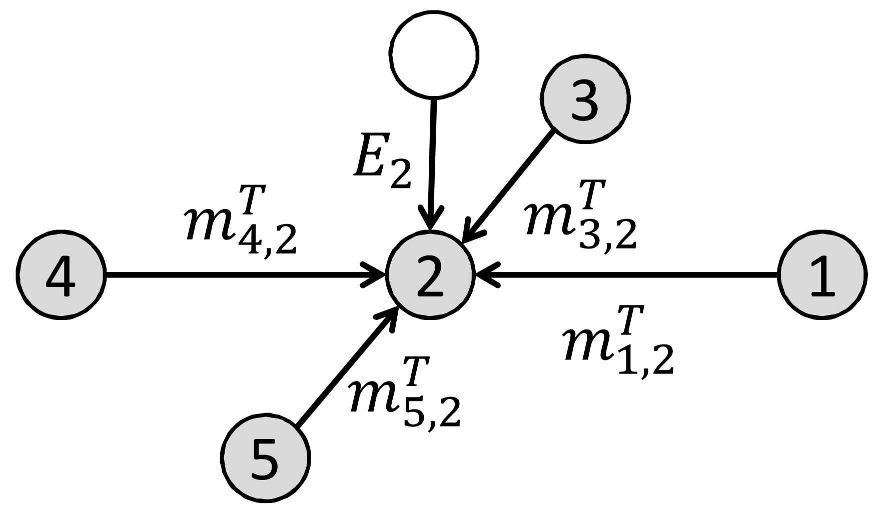
Belief Propagation (BP)

Iterative process in which neighbouring nodes "talk" to each other:

– Update message between neighboring pixels



– Stop after T iterations, decide the final label by picking the smallest dis-belief



- **Challenge**:

When the label set L is huge or densely sampled, BP faces prohibitively high computational challenges.

Particle **Belief Propagation (PBP)**

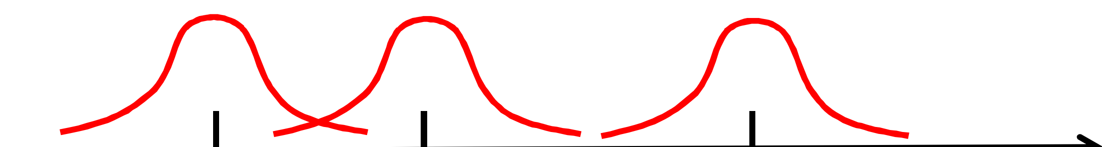
[Ihler and McAllester, "Particle Belief Propagation," *AISTATS*'09]

– **Solution**:

(1) only store messages for *K* labels (particles)

l (discrete label)

(2) generate new label particles with the MCMC sampling using a Gaussian proposal distribution



l

▪**Challenge**:

MCMC sampling is still inefficient and slow for continuous label spaces (e.g. stereo with slanted surfaces).

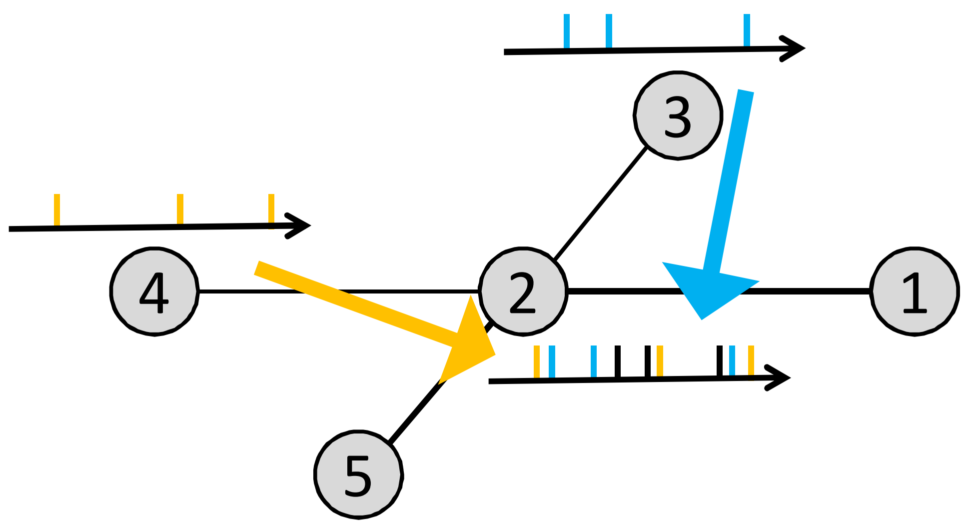
Patch Match Belief Propagation (PMBP)

[Besse et al, "PMBP: PatchMatch Belief Propagation for Correspondence Field Estimation," *IJCV* 2014]

- **Solution**:

Use Patch Match[Barnes et al. Siggraph'09]'s sampling algorithm – augment PBP with label samples from the neighbours as proposals

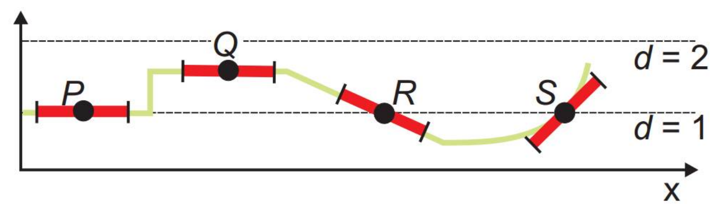
- Orders of magnitude faster than PBP



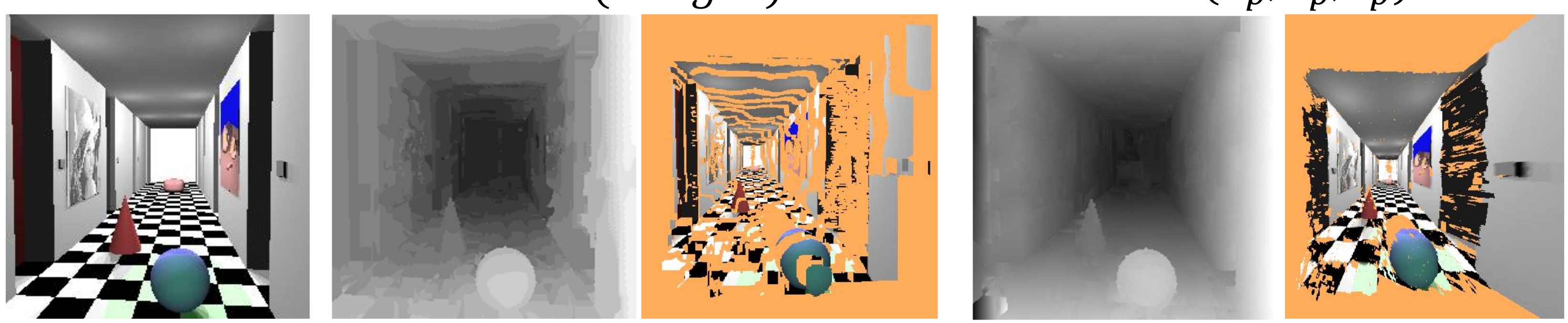
Patch Match Belief Propagation (PMBP)

- Effectively handles large label spaces in message passing
- Successfully applied to stereo with slanted surface modeling [Bleyer et al., BMVC'11]

Label: 3D plane normal $l = (a_p, b_p, c_p)$



$l = d\ (integer)$ $\quad$ $l = (a_p, b_p, c_p)$



Left image | Disparity map | 3D reconstruction | Disparity map | 3D reconstruction

Image courtesy of [Bleyer et al., BMVC'11]

- Also successfully applied to optical flow [Hornáček et al., ECCV'14]

Problem of PMBP

- However, it suffers from a heavy computational load on the data cost computation

$$E = \boxed{\sum_{p} E_p(l_p; W)} + \sum_{p} \sum_{q \in \mathcal{N}_p} E_{pq}(l_p, l_q)$$

- Many works strongly suggest to gather stronger evidence from a local window for the data term

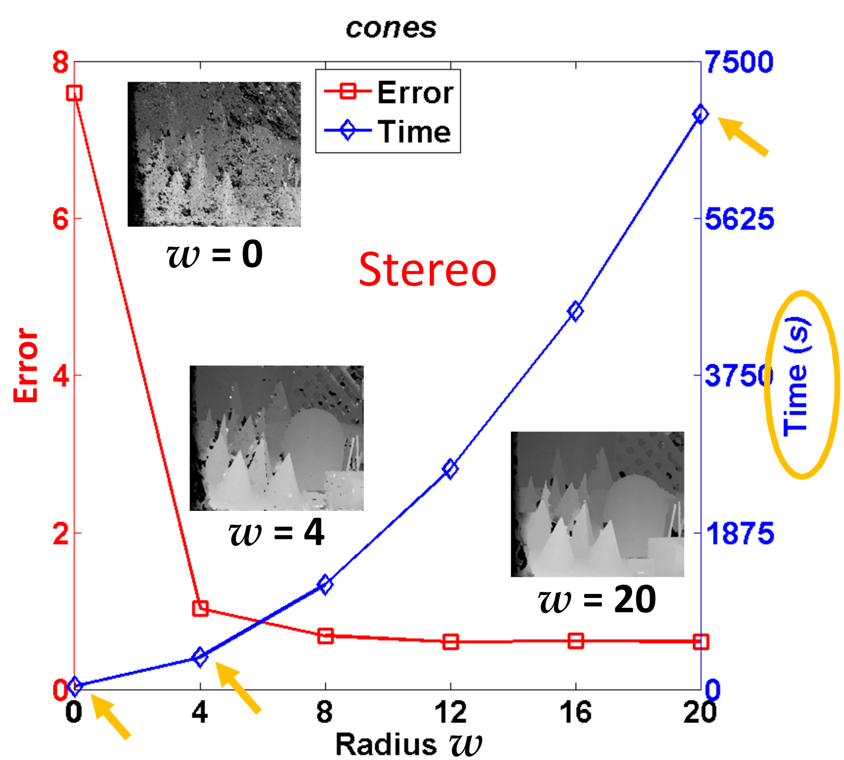
$$E_p(l_p; W) = \sum_{r \in W} \omega_{pr} C_r(l_p)$$

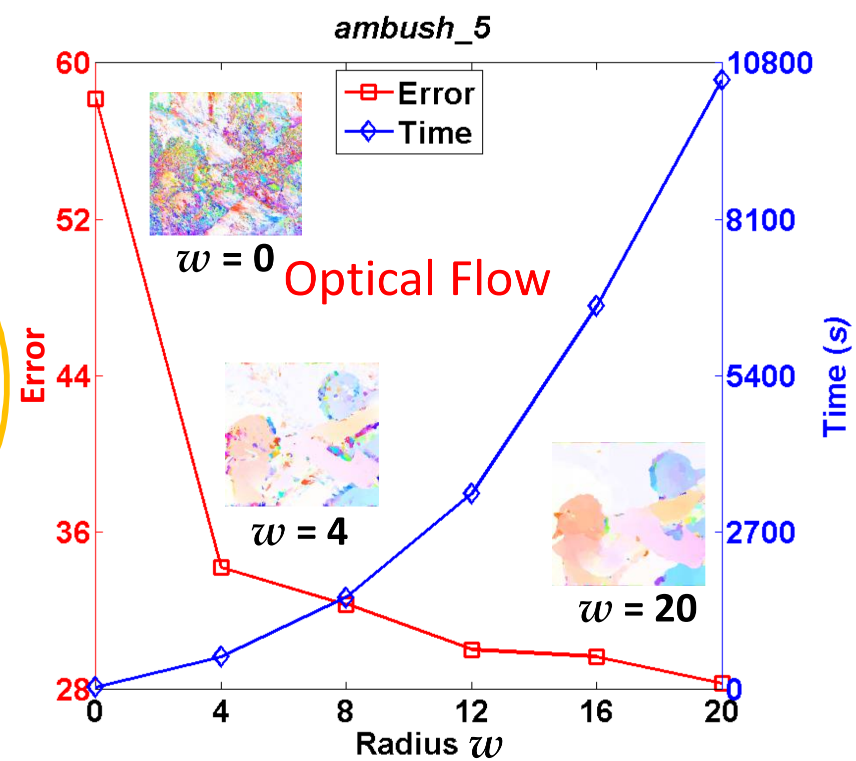
Left view Right view Weight Raw matching cost

Data term is important!

- *Better results with larger window sizes (2w+1)^2, but more computational cost!*

$$\boxed{E_p(l_p; W) = \sum_{r \in W} \omega_{pr} C_r(l_p)}$$





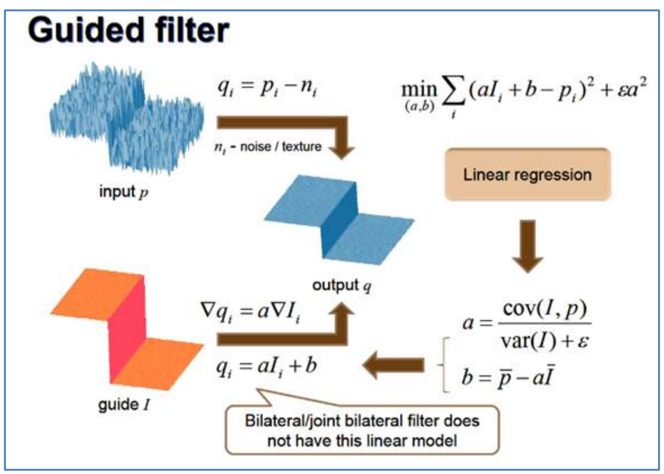
Aggregated data cost computation

- Cross/joint/bilateral filtering principles

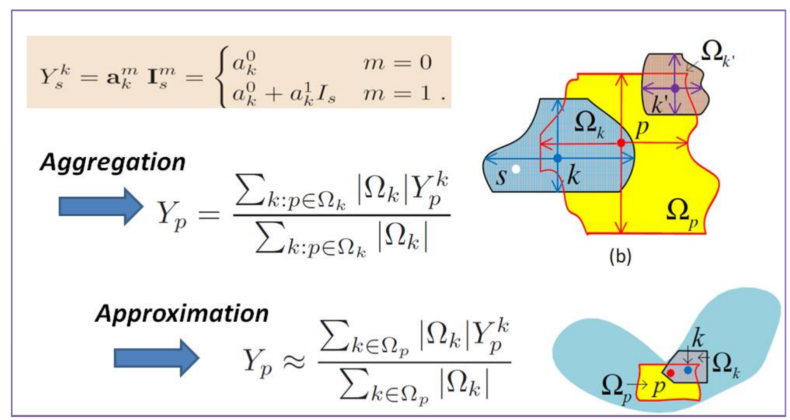
$$E_p(l_p; W) = \sum_{r \in W} \omega_{pr} C_r(l_p)$$

- **Local discrete labeling approaches** have often used efficient O(1)-time edge-aware filtering (EAF) methods [Rhemann et al., CVPR'11].
  - O(1)-time: No dependency on window size used in EAF

Guided Filter [He et al. *ECCV* 2010]



Cross-based Local Multipoint Filtering (CLMF) [Lu et al. *CVPR* 2012]



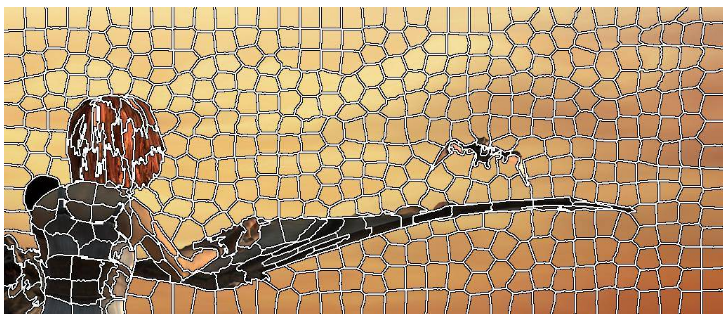
Why does PMBP NOT use O(1) time EAF?

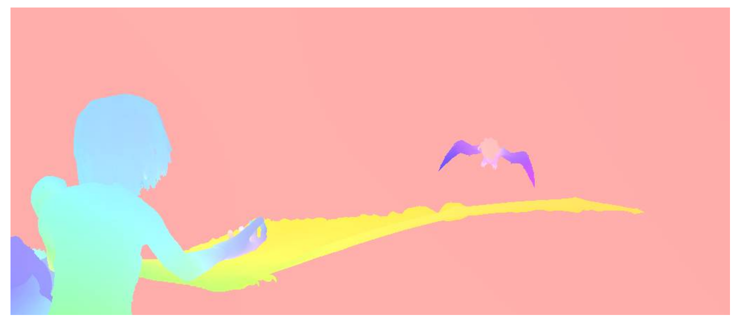
- Particle sampling and data cost computation are performed independently for each pixel

➔ Incompatible with EAF, essentially exploiting redundancy

- **Observation**

Labeling is often spatially smooth away from edges. This allows for shared label proposal and data cost computation for spatially neighboring pixels.





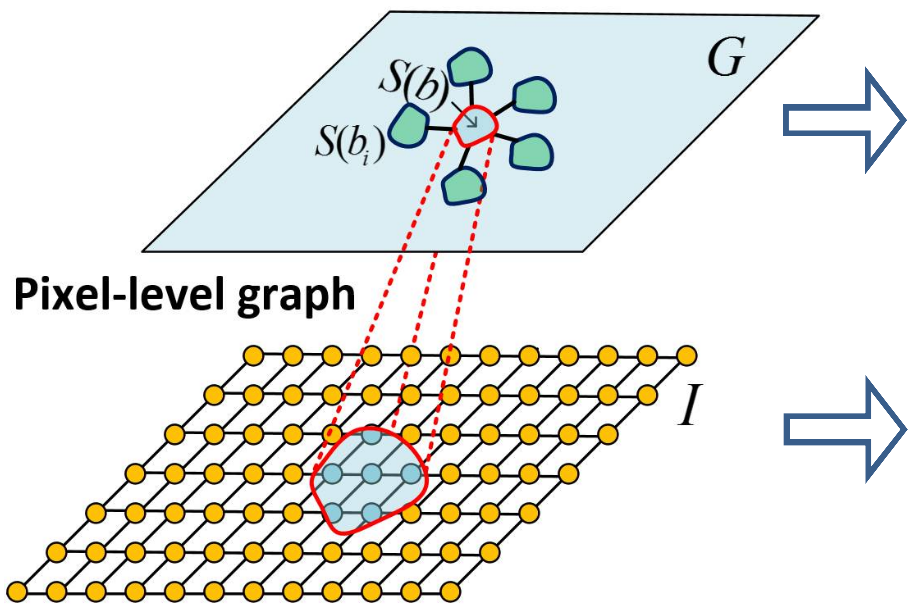
- **Our solution**

A superpixel based particle sampling belief propagation method, leveraging efficient filter-based cost aggregation

Sped-up Patch Match Belief Propagation (SPM-BP)

Sped-up Patch Match Belief Propagation

- **Two-Layer Graph Structures in SPM-BP**



1. Shared particle generation
2. Shared data cost computation

$$E = \boxed{\sum_p E_p(l_p; W)} + \sum_p \sum_{q \in \mathcal{N}_p} E_{pq}(l_p, l_q)$$

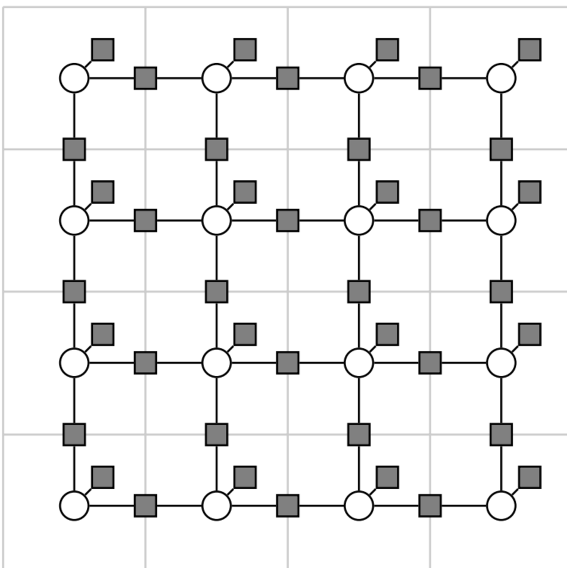
1. Message passing
2. Particle selection

$$E = \sum_p E_p(l_p; W) + \boxed{\sum_p \sum_{q \in \mathcal{N}_p} E_{pq}(l_p, l_q)}$$

- **Scan Superpixels and Perform :**
  - *Neighbourhood Propagation*
  - *Random Search*

Related works

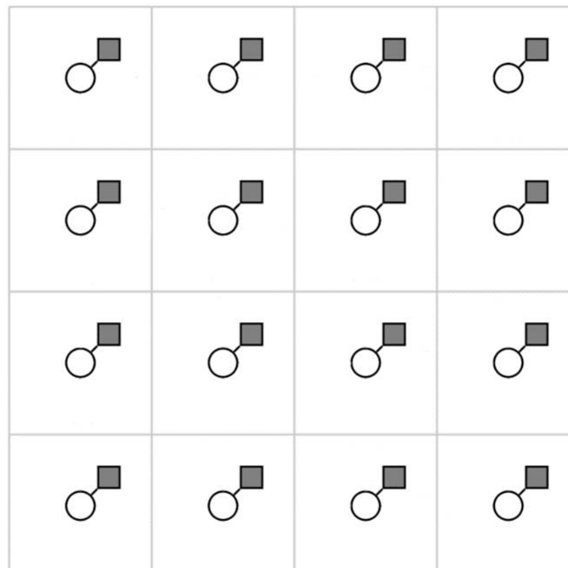
Pixel based MRF



Local methods

[Rhemann et al., CVPR’11]
[Lu et al., CVPR’13]

Only rely on data term

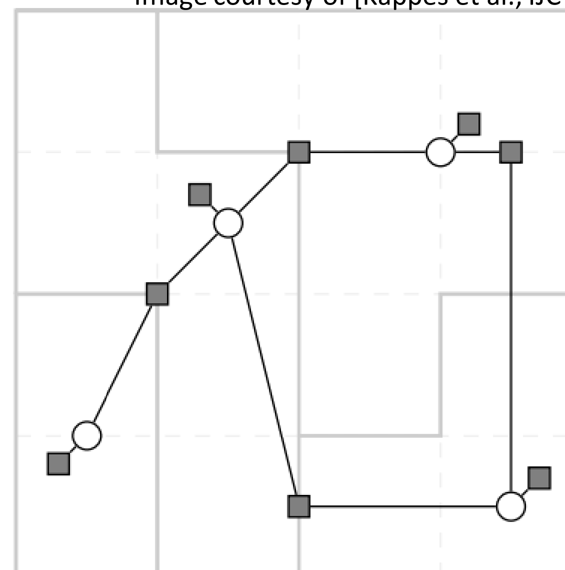


Superpixel based MRF

[Kappes et al., IJCV’15]
[Güney & Geiger, CVPR’15]

Superpixels as graph nodes

Image courtesy of [Kappes et al., IJCV’15]



Superpixel-based MRF: each superpixel is a node in the graph and **all pixels of the superpixel are constrained to have the same label**.

Our two-layer graph: superpixel are employed only for particle generation and data cost computation, the **labeling is performed for each pixel independently**.

Comparison of existing labeling optimizers

Local labeling approaches		Data cost computation	
		w/o EAF: $O(\lvert W \rvert)$	w/ EAF: $O(1)$
Label space handling	w/o PatchMatch: $O(\lvert L \rvert)$		
	w/ PatchMatch: $O(\log \lvert L \rvert)$		

Global labeling approaches		Data cost computation	
		w/o EAF: $O(\lvert W \rvert)$	w/ EAF: $O(1)$
Label space handling	w/o PatchMatch: $O(\lvert L \rvert)$		
	w/ PatchMatch: $O(\log \lvert L \rvert)$		

Comparison of existing labeling optimizers

Local labeling approaches		Data cost computation	
		w/o EAF: $O(\|W\|)$	w/ EAF: $O(1)$
Label space handling	w/o PatchMatch: $O(\|L\|)$	Adaptive Weighting [PAMI'06]	Cost Filtering [CVPR'11]
	w/ PatchMatch: $O(\log\|L\|)$	PM Stereo [BMVC'11]	**PMF** [CVPR'13]

Global labeling approaches		Data cost computation	
		w/o EAF: $O(\|W\|)$	w/ EAF: $O(1)$
Label space handling	w/o PatchMatch: $O(\|L\|)$		
	w/ PatchMatch: $O(\log\|L\|)$		

Comparison of existing labeling optimizers

<table>
<tr><td colspan="2" rowspan="2">Local labeling approaches</td><td colspan="2">Data cost computation</td></tr>
<tr><td>w/o EAF: O(|W|)</td><td>w/ EAF: O(1)</td></tr>
<tr><td rowspan="2">Label space handling</td><td>w/o PatchMatch: O(|L|)</td><td>Adaptive Weighting [PAMI’06]</td><td>Cost Filtering [CVPR’11]</td></tr>
<tr><td>w/ PatchMatch: O(log|L|)</td><td>PM Stereo [BMVC’11]</td><td>PMF [CVPR’13]</td></tr>
</table>

<table>
<tr><td colspan="2" rowspan="2">Global labeling approaches</td><td colspan="2">Data cost computation</td></tr>
<tr><td>w/o EAF: O(|W|)</td><td>w/ EAF: O(1)</td></tr>
<tr><td rowspan="2">Label space handling</td><td>w/o PatchMatch: O(|L|)</td><td>BP [PAMI’06]</td><td>Fully-connected CRFs [NIPS’11]</td></tr>
<tr><td>w/ PatchMatch: O(log|L|)</td><td>PMBP [IJCV’14]</td><td>?</td></tr>
</table>

Comparison of existing labeling optimizers

Local labeling approaches		Data cost computation	
		w/o EAF: O(|W|)	w/ EAF: O(1)
Label space handling	w/o PatchMatch: O(|L|)	Adaptive Weighting [PAMI'06]	Cost Filtering [CVPR'11]
	w/ PatchMatch: O(log|L|)	PM Stereo [BMVC'11]	**PMF** [CVPR'13]

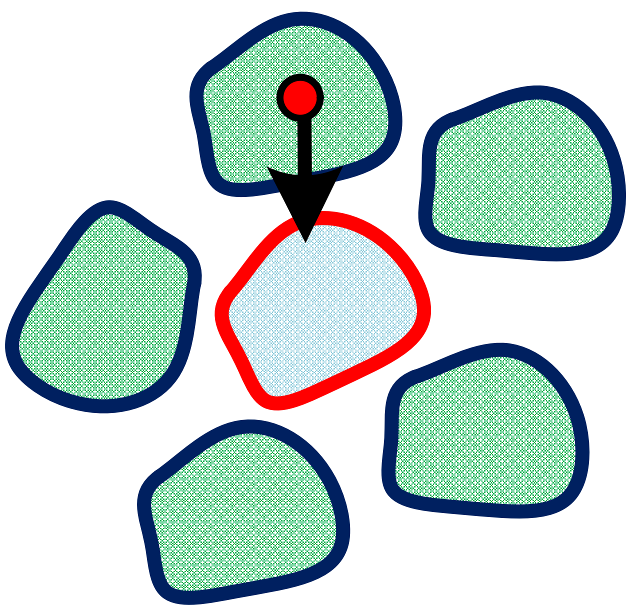
Global labeling approaches		Data cost computation	
		w/o EAF: O(|W|)	w/ EAF: O(1)
Label space handling	w/o PatchMatch: O(|L|)	BP [PAMI'06]	Fully-connected CRFs [NIPS'11]
	w/ PatchMatch: O(log|L|)	**PMBP** [IJCV'14]	**SPM-BP [This paper]**

SPM-BP: Neighbourhood Propagation

- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update

1-1) Randomly select one pixel from each neighbouring superpixel

1-2) Add the particles at these pixels into the proposal set



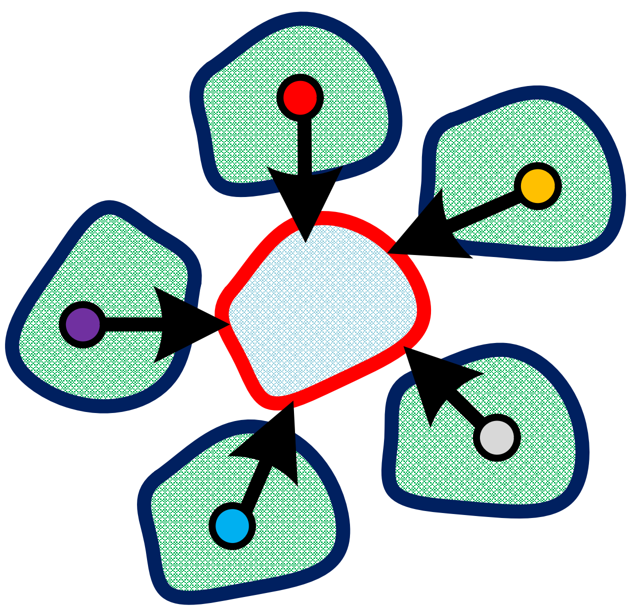
K=3

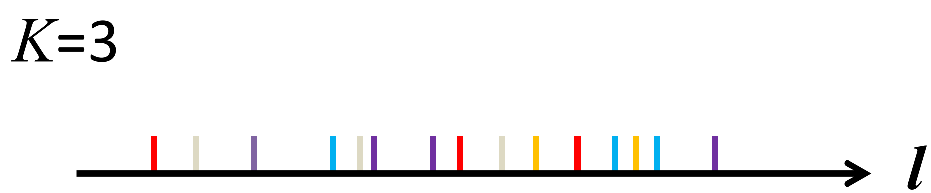
SPM-BP: Neighbourhood Propagation

- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update

1-1) Randomly select one pixel from each neighbouring superpixel

1-2) Add the particles at these pixels into the proposal set

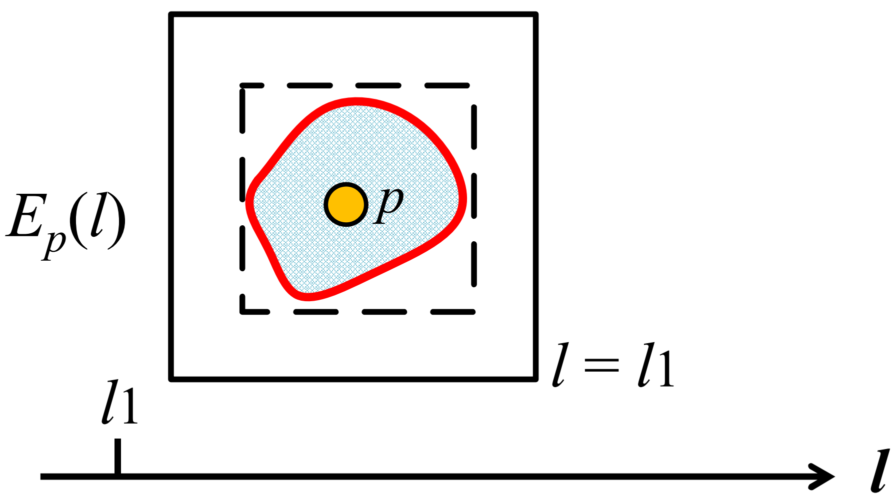




SPM-BP: Neighbourhood Propagation

- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update

2-1) Compute the raw matching data cost of these labels in a slightly enlarged region
2-2) Compute the aggregated data cost for each label by performing EAF on the raw matching cost

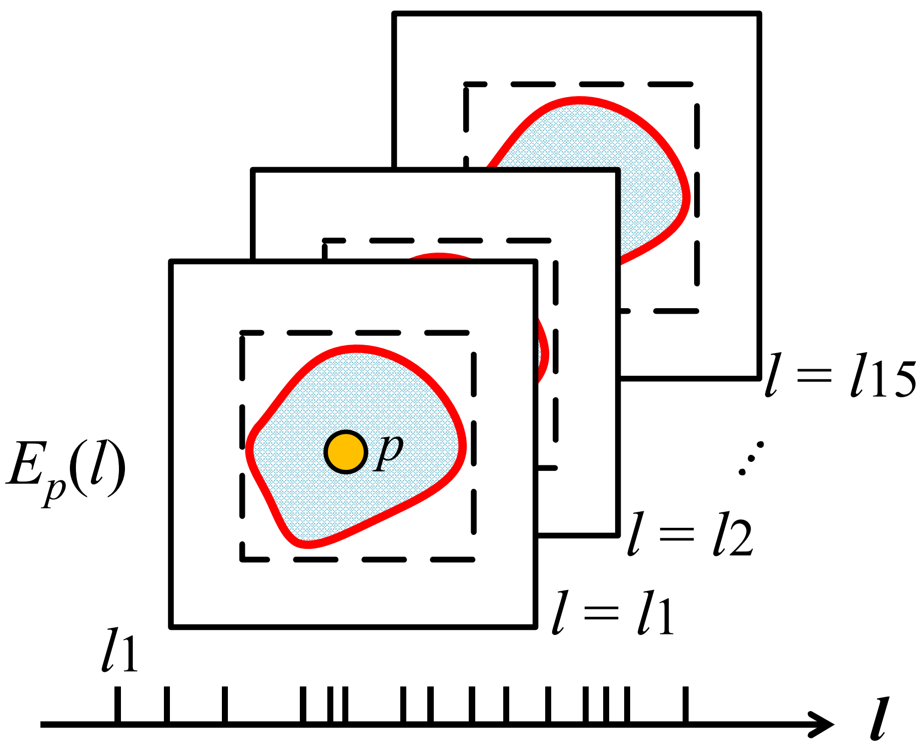


SPM-BP: Neighbourhood Propagation

- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update

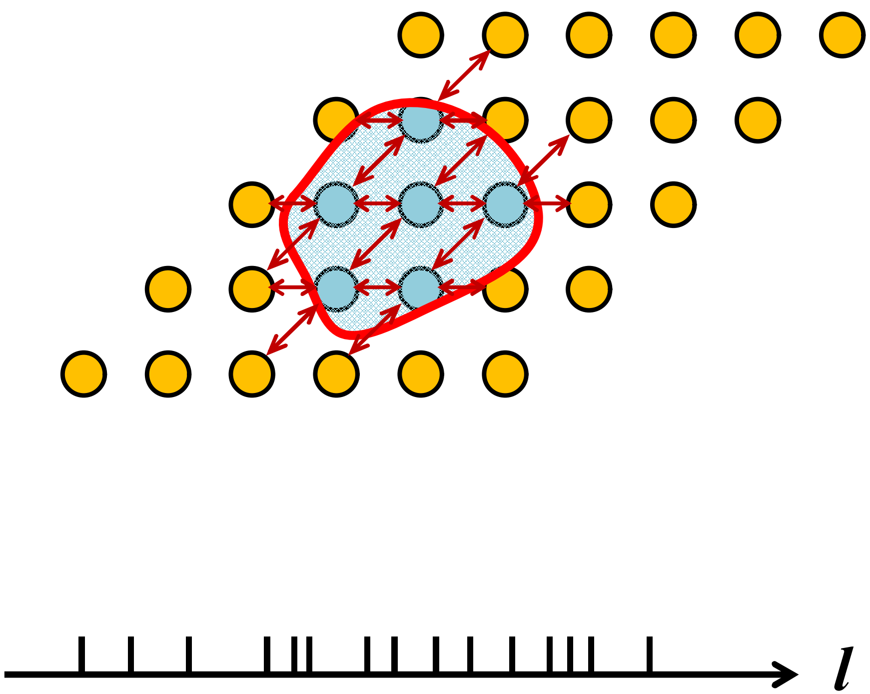
2-1) Compute the raw matching data cost of these labels in a slightly enlarged region

2-2) Compute the aggregated data cost for each label by performing EAF on the raw matching cost



SPM-BP: Neighbourhood Propagation

- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update



3-1) Perform message passing for pixels within the superpixel.

SPM-BP: Neighbourhood Propagation

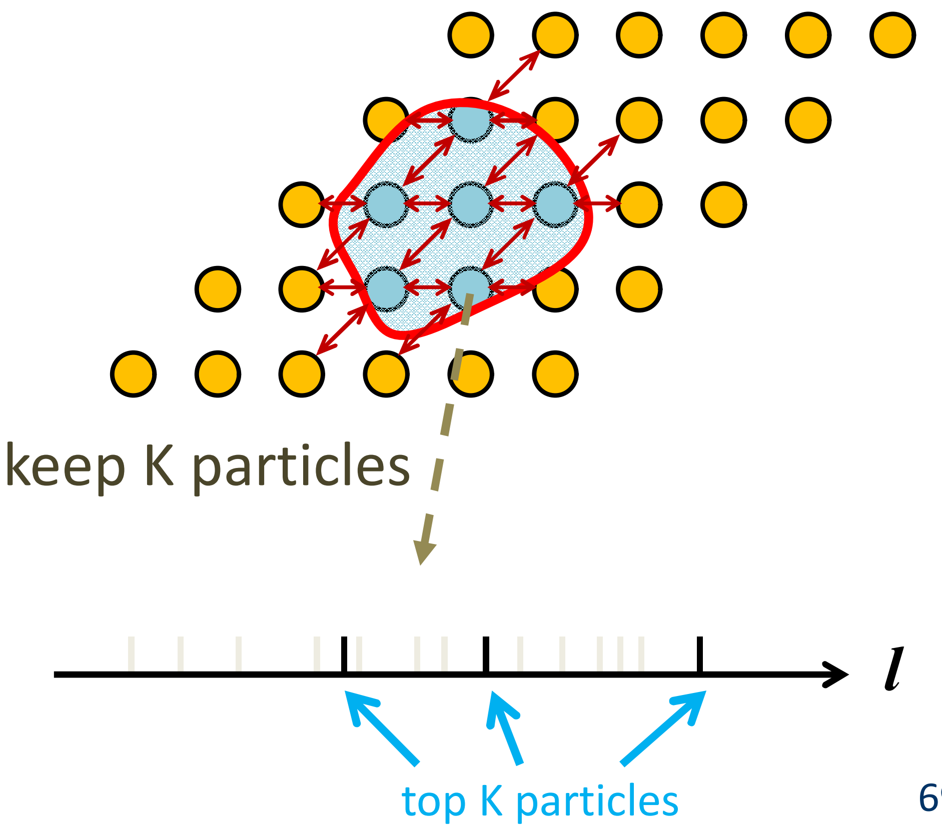
✓ Step 1. Particle propagation

✓ Step 2. Data cost computation

✓ Step 3. Message update

3-1) Perform message passing for pixels within the superpixel.

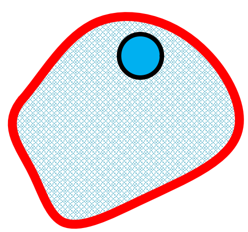
3-2) Keep *K* particles with the smallest disbeliefs at each pixel.



SPM-BP: Random Search

- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update

1-1) Randomly select one pixel in the visiting superpixel

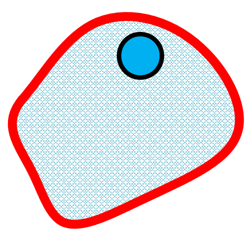


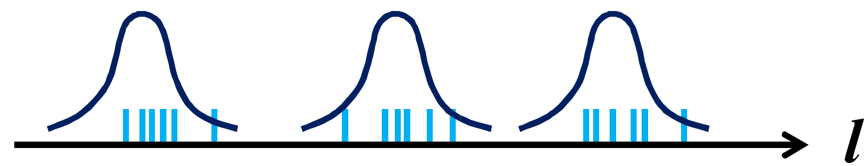
SPM-BP: Random Search

- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update

1-1) Randomly select one pixel in the visiting superpixel

1-2) Generate new proposals around the sampled particles





SPM-BP: Random Search

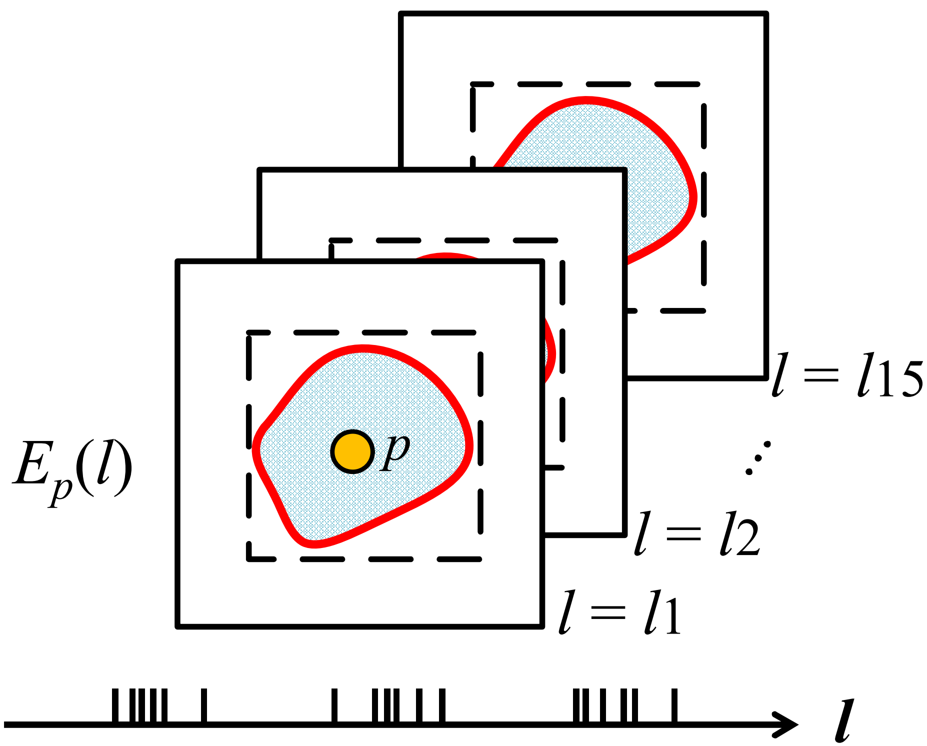
✓ Step 1. Particle propagation

✓ Step 2. Data cost computation

✓ Step 3. Message update

2-1) Compute the raw matching data cost of these labels in a slightly enlarged region

2-2) Compute the aggregated data cost for each label by performing EAF on the raw matching cost

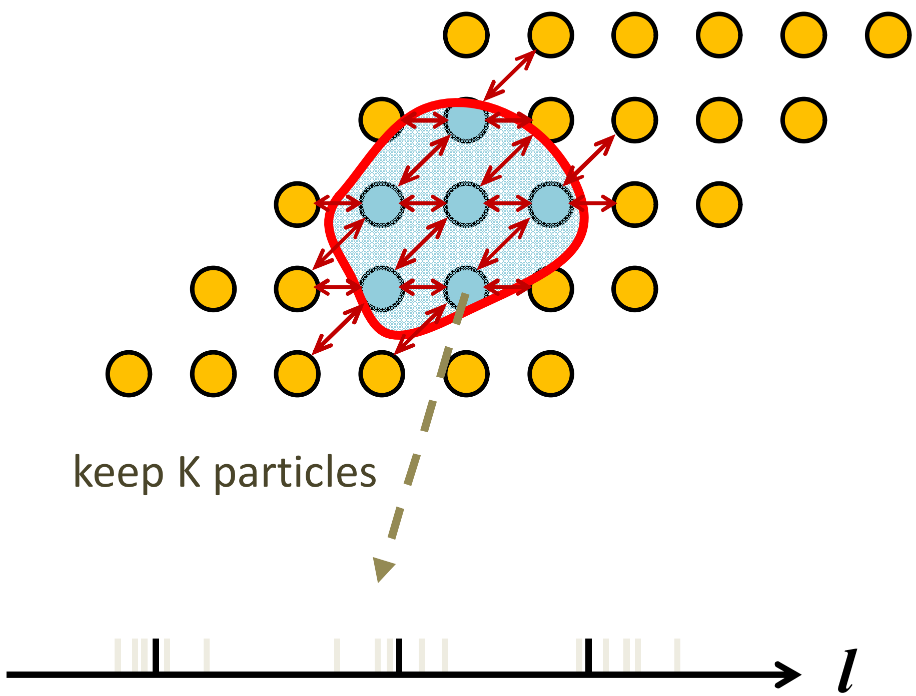


$$E_p(l_p; W) = \sum_{r \in W} \omega_{pr} C_r(l_p)$$

SPM-BP: Random Search

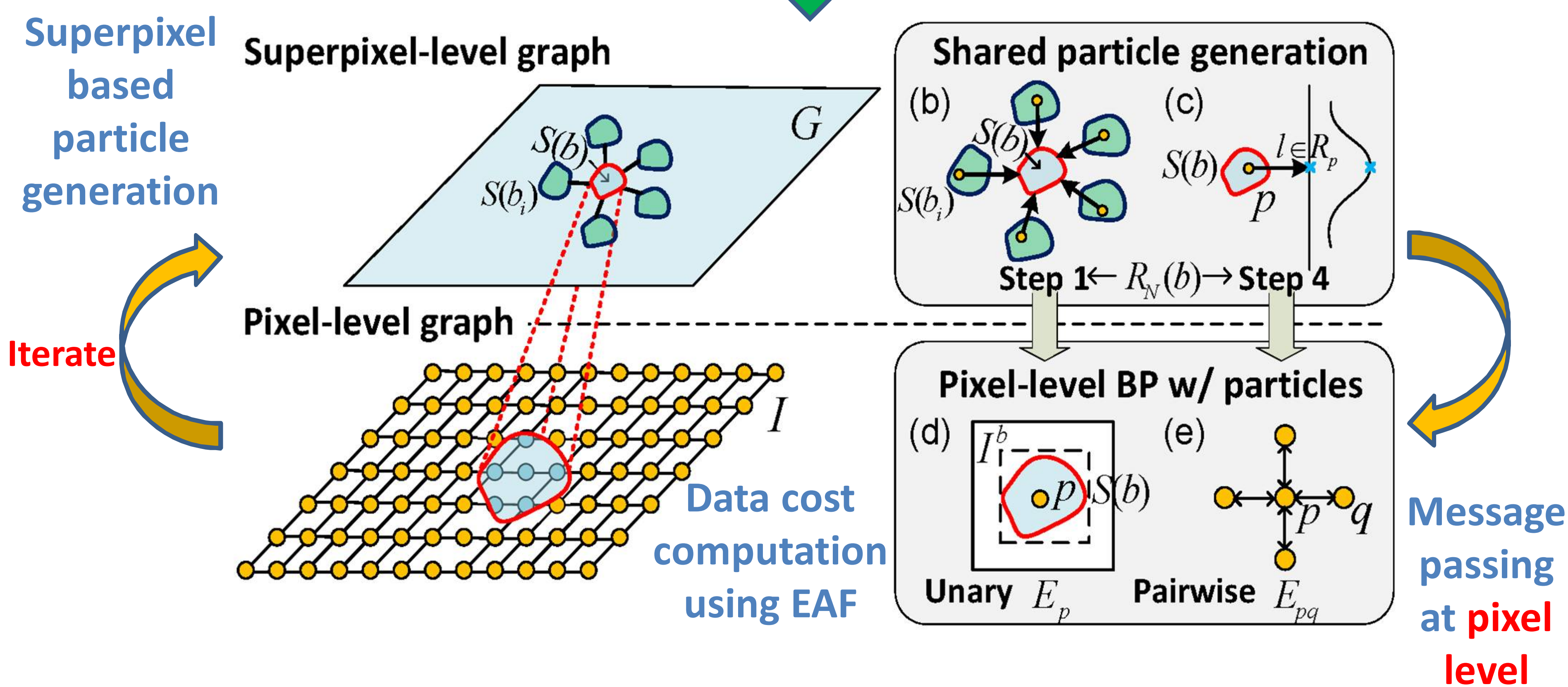
- ✓ Step 1. Particle propagation
- ✓ Step 2. Data cost computation
- ✓ Step 3. Message update

3-1) Perform message passing for pixels within the superpixel.
3-2) Keep *K* particles with the smallest disbeliefs at each pixel.



SPM-BP: Recap

Random Initialization



Final labels

Complexity Comparison

	PMF* [32]	PMBP [8]	**SPM-BP**
Data Cost	$O(N\log L)$	$O(\lvert W\rvert K N\log L)$	$O(KN\log L)$
Message Passing	-	$O(K^2 N\log L)$	$O(K^2 N\log L)$

$|W|$ – local window size (e.g. 31x31 for stereo)

K – number of particles used (small constant)

N – number of pixels

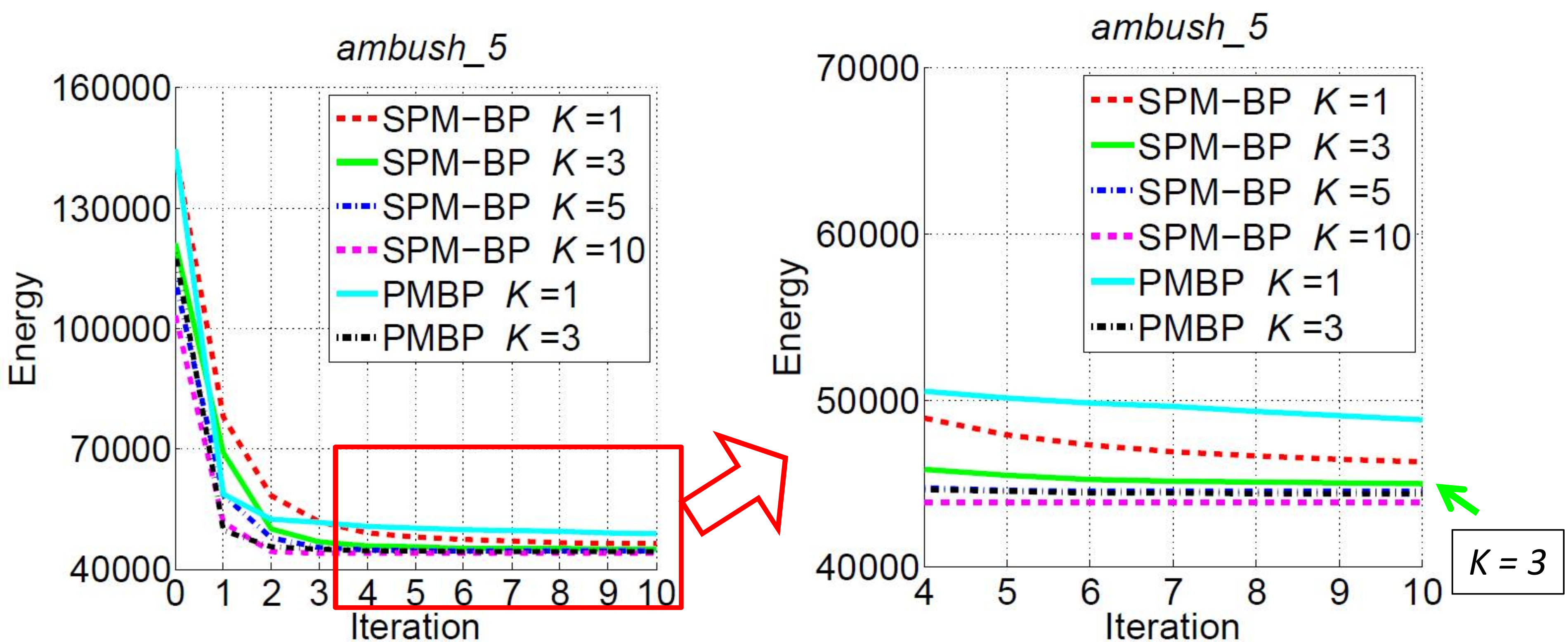
L – label space size (e.g. over 10 million for flow)

*PMF stores only one best particle (K = 1) per pixel node, thus requiring more iterations than the other two methods.

Example Applications

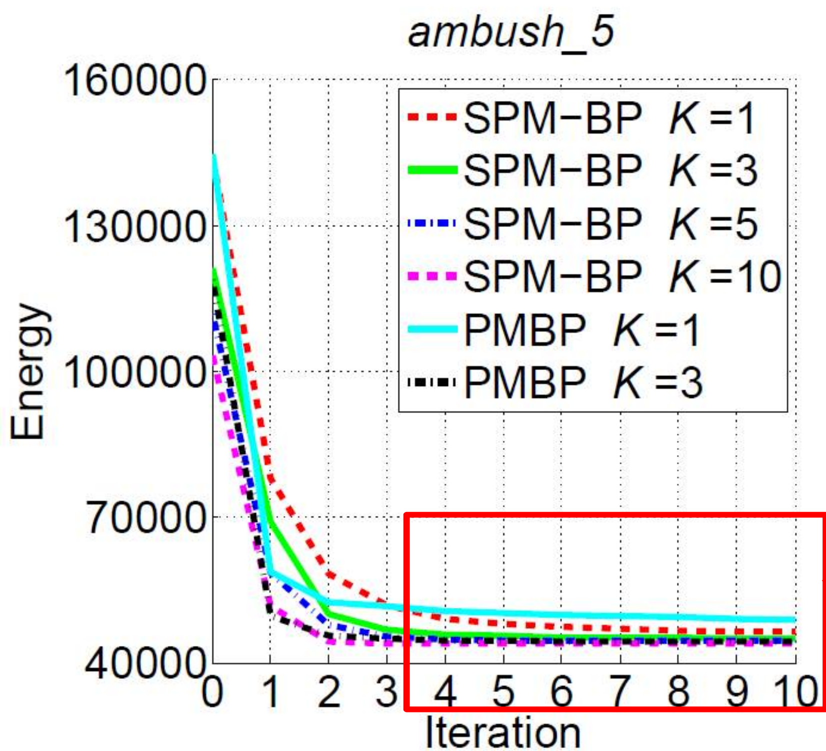
- **Stereo with slanted surface supports**
  - **label**: 3D plane normal $l_p = (a_p, b_p, c_p)$
  - **Matching features**: color + gradient
  - **Smoothness term**: deviation between two local planes
  - **Cross checking + post processing for occlusion**
- **Large-displacement optical flow**
  - **label**: 2D displacement vector $l_p = (u, v)$
  - **Matching features**: color + Census transform
  - **Smoothness term**: truncated L_2 distance
  - **Cross checking + post processing for occlusion**

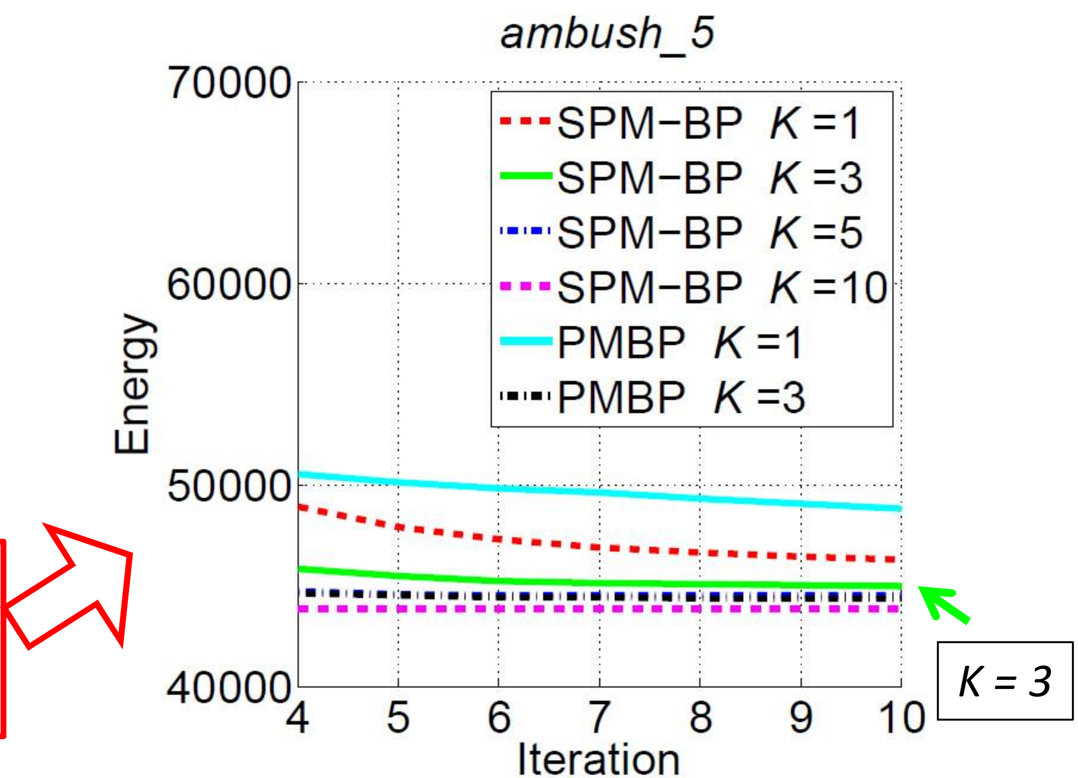
Convergence

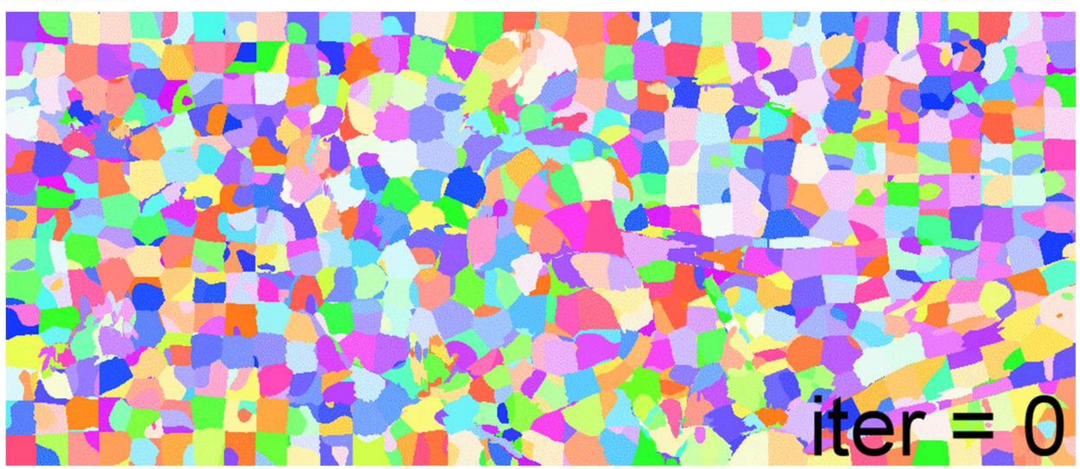


#iteration = 5, K = 3

Convergence

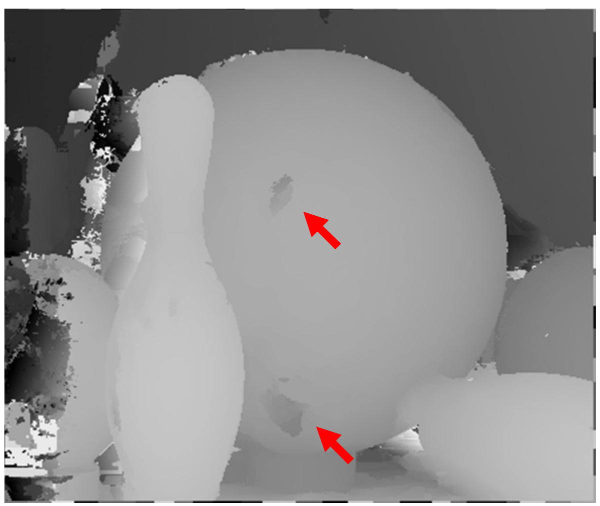




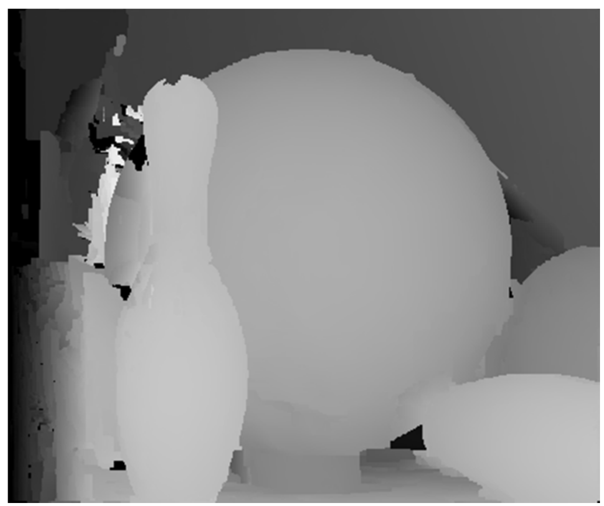


Stereo results

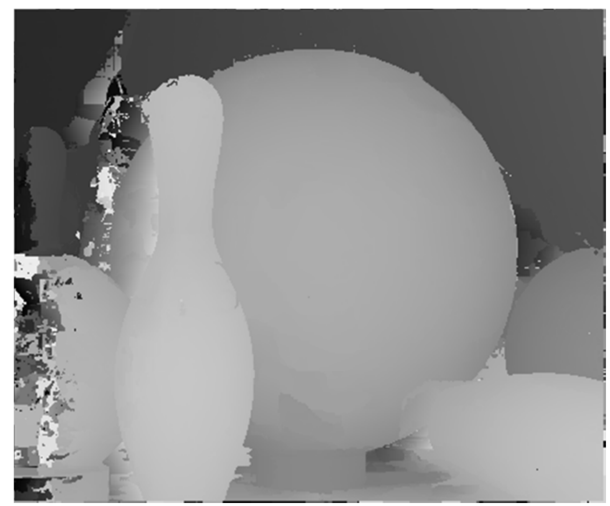
Stereo input



PMF
20 sec.



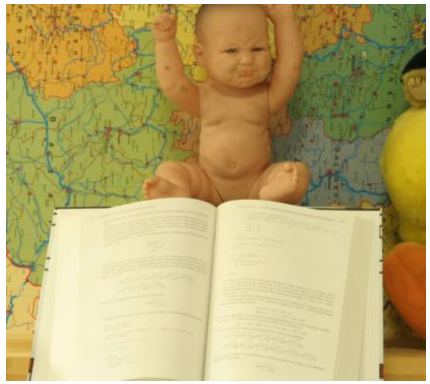
PMBP
3100 sec.

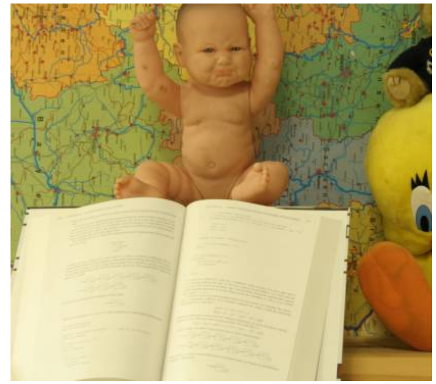


SPM-BP (ours)
30 sec.

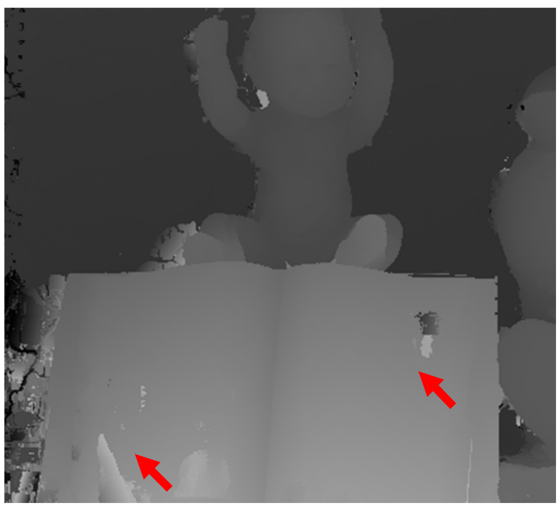
Much faster than PMBP, and much better than PMF for textureless regions

Stereo results



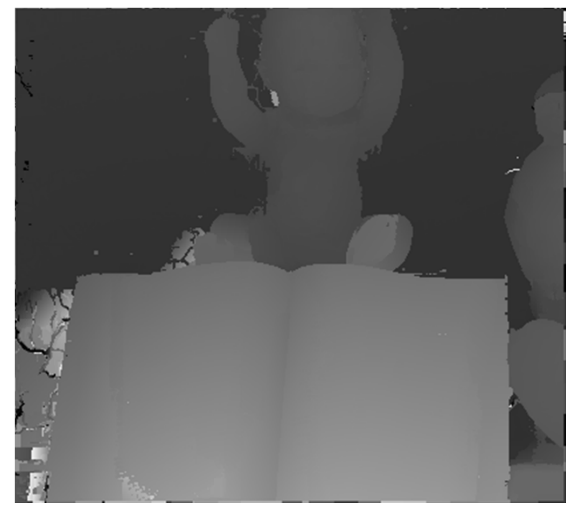


Stereo input



PMF
20 sec.

PMBP
3100 sec.



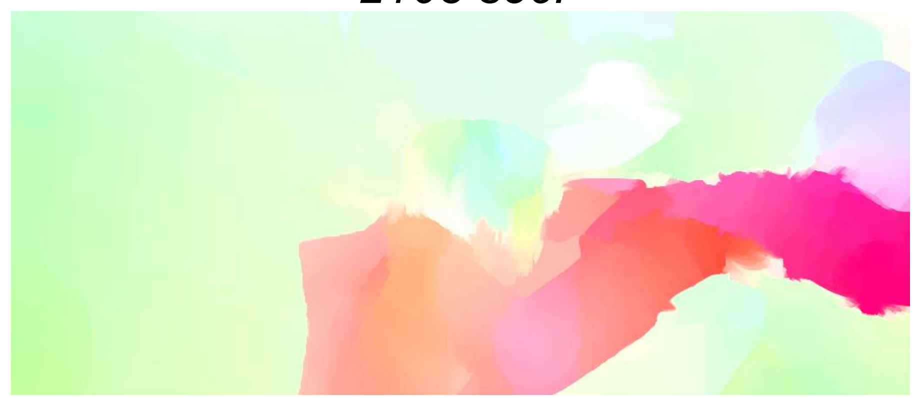
SPM-BP (ours)
30 sec.

Optical flow results

Optical flow input

PMBP
2103 sec.

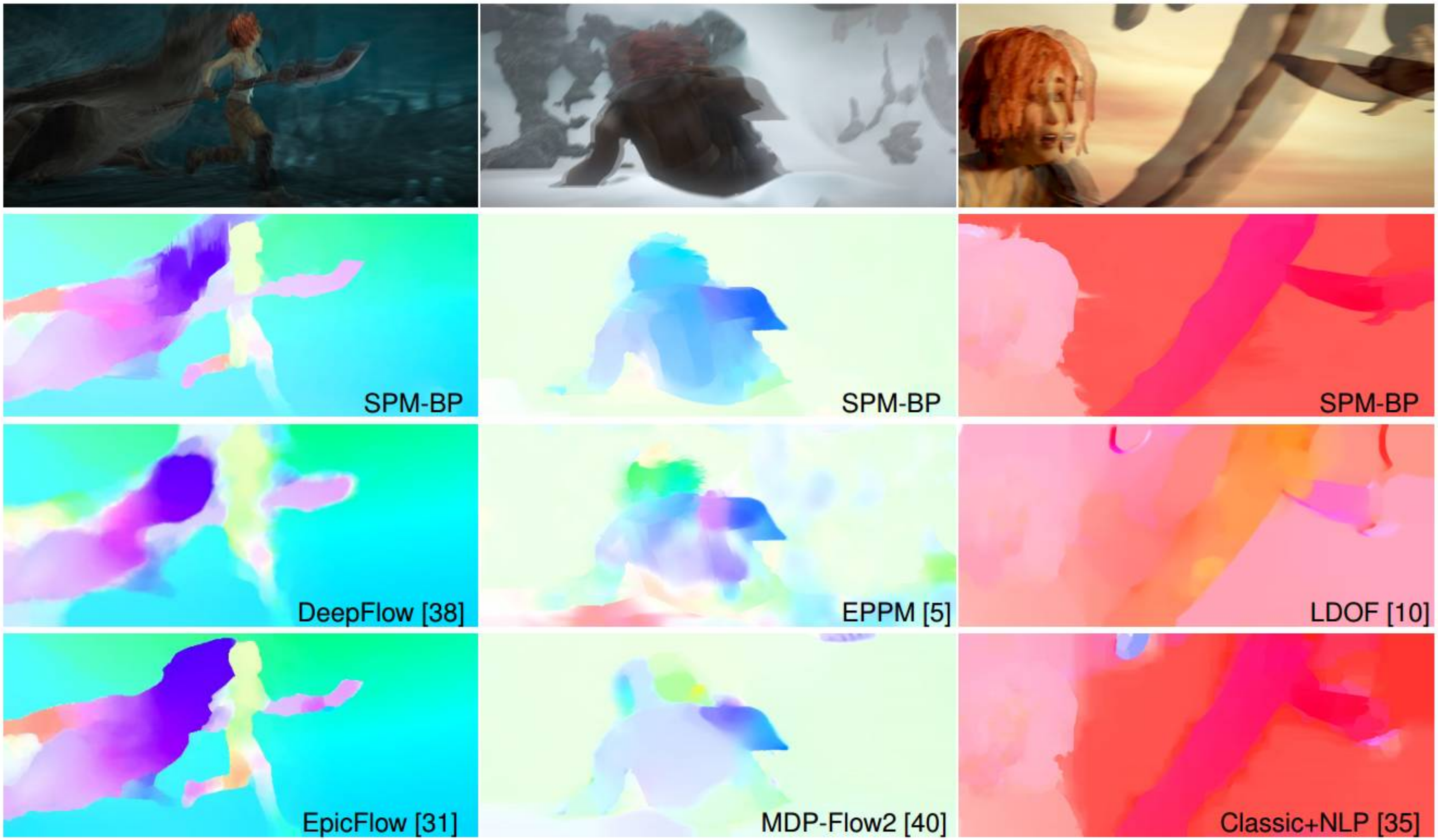
PMF
27 sec.



SPM-BP (ours)
42 sec.

Much faster than PMBP, and much better than PMF for textureless regions

Optical flow results



Performance Evaluation

Middlebury Stereo Performance (Tsukuba/Venus/Teddy/Cones)

Method	Avg. Rank	Avg. Error	Runtime(s)
PM-PM [39]	8.2	7.58	34 (GPU)
PM-Huber [17]	8.4	7.33	52 (GPU)
SPM-BP	12.1	7.71	30
PMF [24]	12.3	7.69	20
PMBP [7]	19.8	8.77	3100

Middlebury Stereo 2006 Performance

Dataset	PMF [25]	PMBP [7]	SPM-BP
Baby2	15.34	16.85	**12.82**
Books	**22.15**	27.57	22.52
Bowling2	15.95	15.20	**14.35**
Flowerpots	**24.59**	27.97	24.80
Lampshade1	25.02	30.22	**23.39**
Laundry	**26.77**	33.90	27.32
Moebius	21.47	25.09	**21.09**
Reindeer	**15.04**	21.57	16.02
Mean	20.79	24.79	**20.29**

Optical Flow Performance on MPI Sintel Benchmark (captured on 16/04/2015)

Method	EPE all Clean	Rank	EPE all Final	Rank	Runtime (Sec)
EpicFlow [30]	4.115	1	6.285	1	17
PH-Flow [41]	4.388	2	7.423	8	800
SPM-BP	5.202	5	7.325	6	42
DeepFlow [36]	5.377	7	7.212	4	19
LocalLayering [33]	5.820	13	8.043	13	-
MDP-Flow2 [38]	5.837	14	8.445	21	754
EPPM [5]	6.494	18	8.377	20	0.95*
S2D-Matching [21]	6.510	19	7.872	10	2000
Classic+NLP [34]	6.731	21	8.291	19	688
Channel-Flow [32]	7.023	24	8.835	26	>10000
LDOF [10]	7.563	25	9.116	28	30

Remarks

- A simple formulation, without needing *complex* energy terms nor a separate *initialization*
- Achieved top-tier performance, even when compared to *task-specific* techniques
- Applied on the full pixel grid, avoiding *coarse-to-fine* steps

Conclusion

- SPM-BP is simple, effective and efficient
- Takes the best computational advantages of
  - **efficient edge-aware cost filtering**
  - and **superpixel-based particle-sampling for message passing**
- Offers itself as a general and efficient global optimizer for continuous MRFs
- Future work
  - *Robust* dense correspondences for cross-scene matching
  - Dealing with *high-order* terms in MRF

Code is now available online:

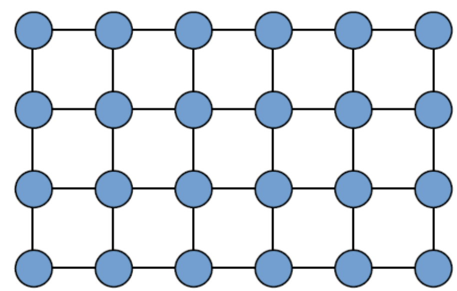
https://github.com/yu-li/spm-bp

Future work (1/2)

- **A better and efficient optimizer for MRF model**
  - Efficient, global discrete optimization for more flexible energy formulation

1. Dealing with **high-dimensional label spaces**
2. Using **stronger unary term** with learning based or cost aggregation based approaches
3. Solving **high-order MRF model**

$$E(\mathbf{x}) = \sum_i \underbrace{\psi_u(x_i)}_{\text{unary term}} + \sum_i \sum_{j>i} \underbrace{\psi_p(x_i, x_j)}_{\text{pairwise term}}$$



Future work (2/2)

- **Recent papers encouraging further research**
  - Sparse2Dense [EpicFlow-CVPR’15]
  - Learning-based regularization [data-driven-3DV’14]
  - Object-level constraint in the regularization [Displets-CVPR’15]

Resources

- ICME'15 tutorial: ***Visual Correspondences: Taxonomy, Modern Approaches and Ubiquitous Applications***

  http://www.icme2015.ieee-icme.org/tutorials.php

  Project page is now available, including codes, slides, and references!
  https://sites.google.com/site/icme15tutorial/

- More resources
  - VMA site (papers, demos, code)
    http://publish.illinois.edu/visual-modeling-and-analytics/
  - CVLAB at CNU
    http://cvlab.cnu.ac.kr/