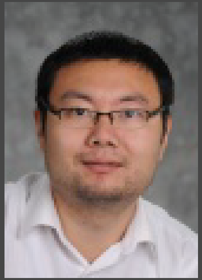
Georgia Tech | Institute for Robotics and Intelligent Machines

Computer Vision and Pattern Recognition 2014

Ahmad Humayun

Fuxin Li



Jim Rehg

RIGOR: Reusing Inference in Graph Cuts for generating Object Regions

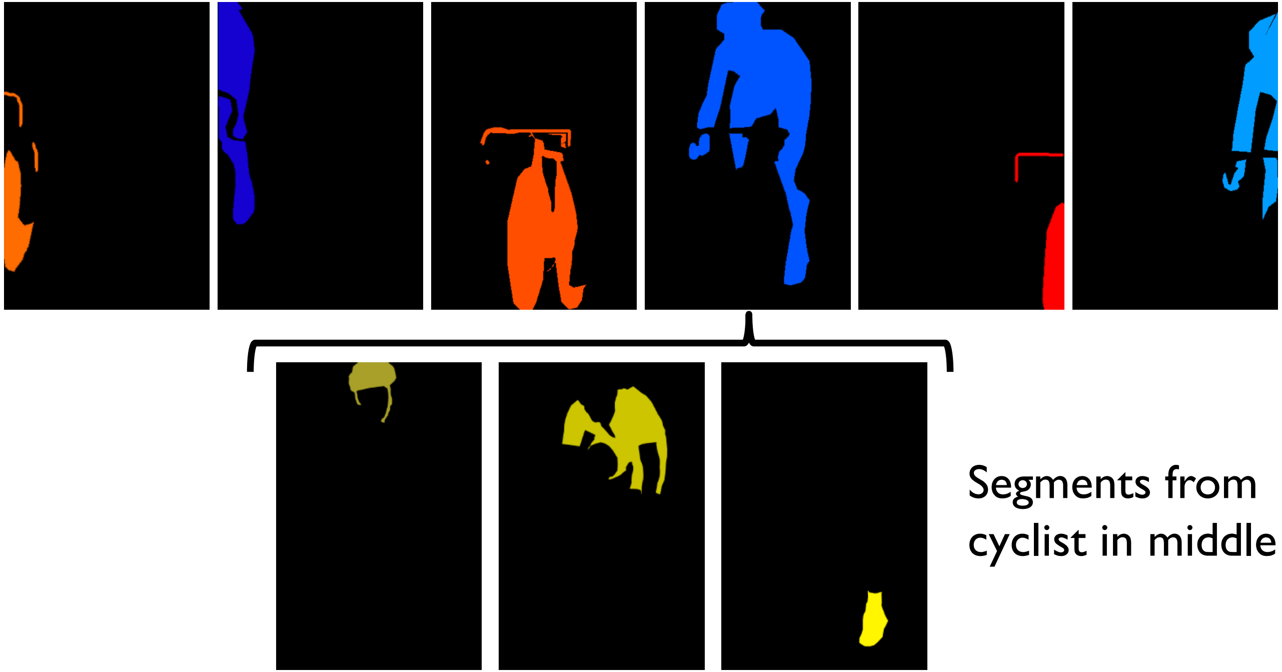
July 2014

http://cpl.cc.gatech.edu/projects/RIGOR

Problem Statement - *Finding Figure-Ground Segments*

Input Image, I

Hierarchy of Object Segments



Segments from cyclist in middle

[1] All input images from PASCAL VOC – Everingham *et al.*, "The PASCAL VOC Challenge," IJCV 2010
[2] Arbeláez *et al.*, "Contour Detection and Hierarchical Image Segmentation," PAMI 2011

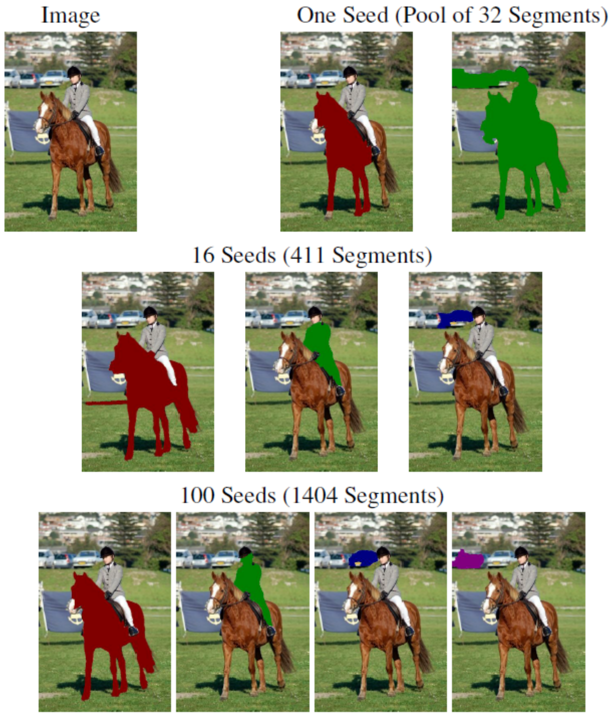
Motivation

How to find objects?

If there is an object at a small selected location (seed) – what is the best segment

Current methods too slow …

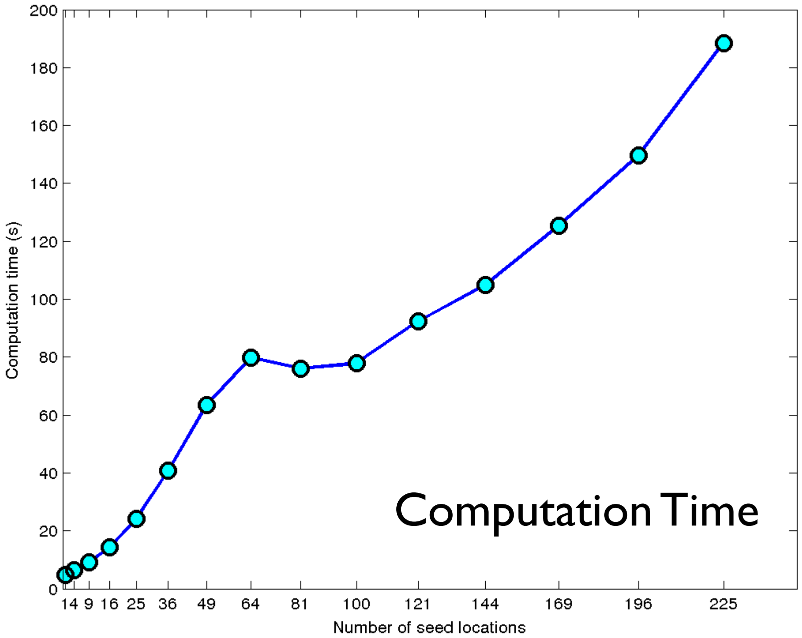
Method	Run Time (s)
CPMC [1]	34.01
Object Proposals [2]	126.46
Shape Sharing [3]	410.31

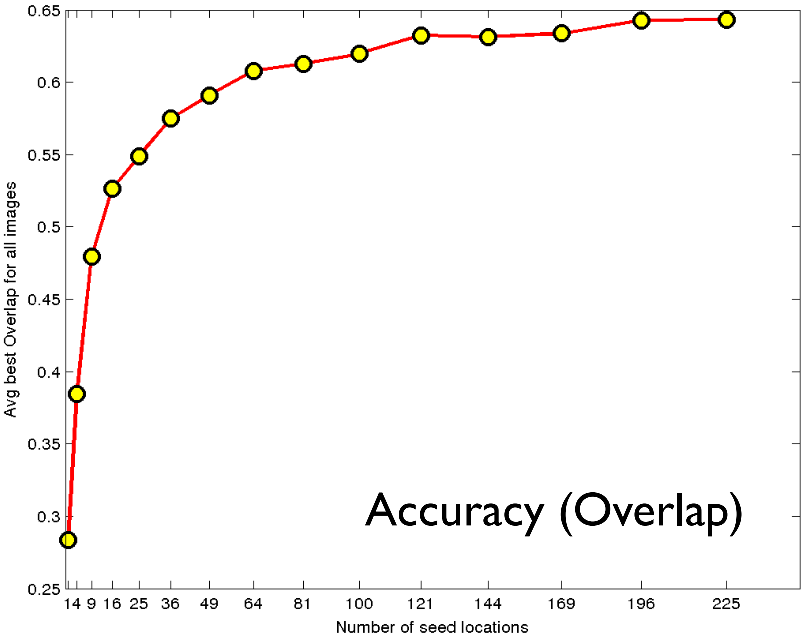


[1] Carreira and Sminchisescu, "CPMC: Automatic Object Segmentations Using Constrained Parametric Min-Cuts," PAMI 2012
[2] Endres and Hoiem, "Category-Independent Object Proposals with Diverse Ranking," PAMI 2013
[3] Kim and Grauman, "Shape Sharing for Object Segmentation," ECCV 2012

Motivation

CPMC: More segments ➔ Slower speed ➔ Higher recall

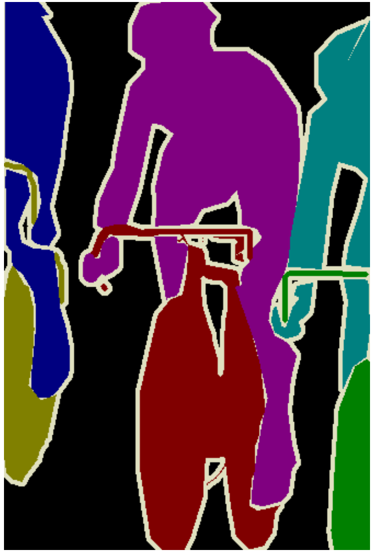




[1] Carreira and Sminchisescu, "CPMC: Automatic Object Segmentations Using Constrained Parametric Min-Cuts," PAMI 2012

Goal

Segments of similar quality in an ***order of magnitude*** less time …

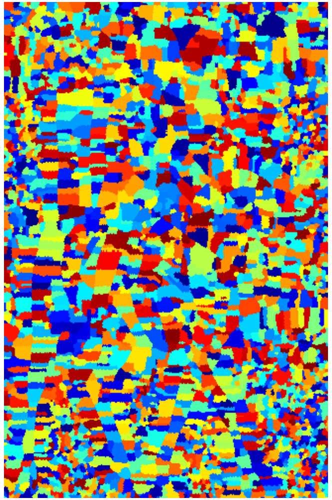
Input Image, I	PASCAL VOC Ground-Truth	RIGOR best object proposals
		

Method Overview

Input Image, I

Probabilistic Boundaries [1,2,3]

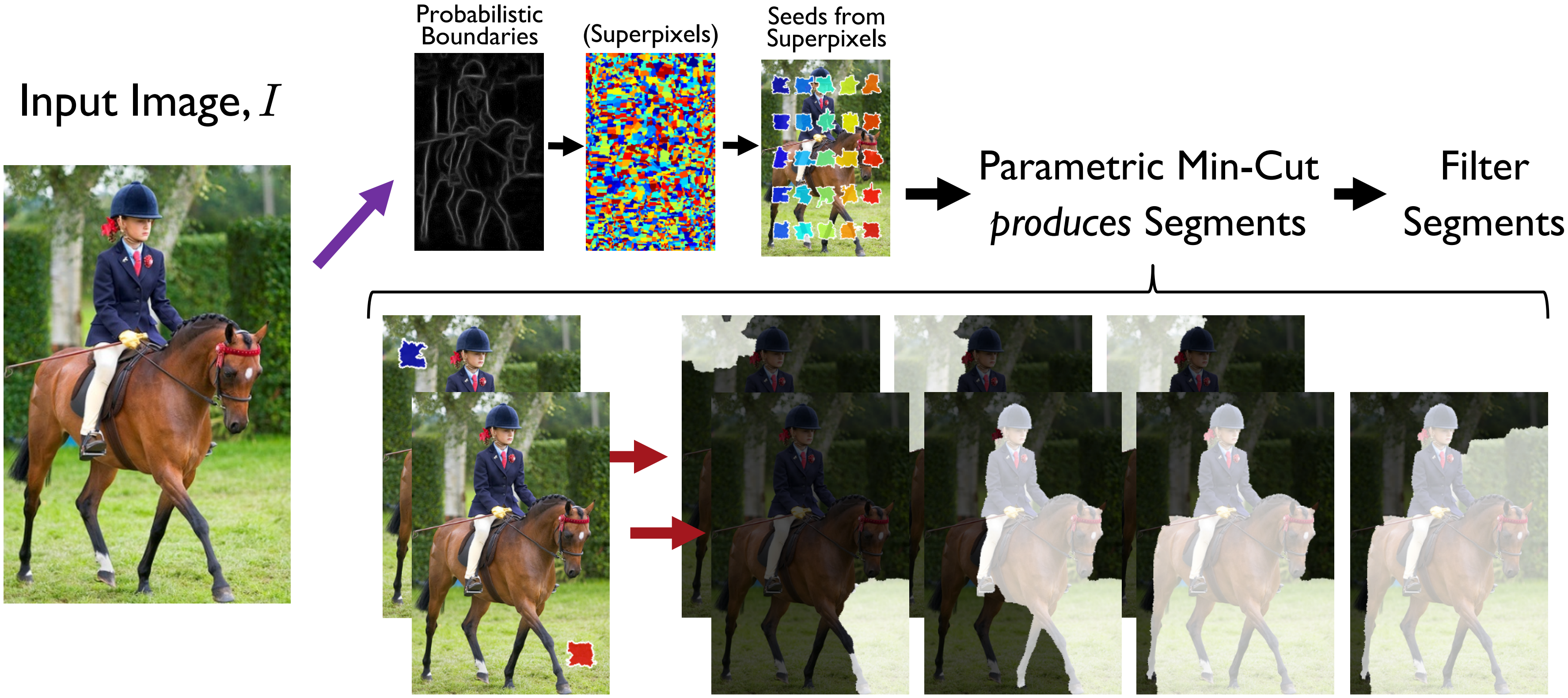
(Superpixels)



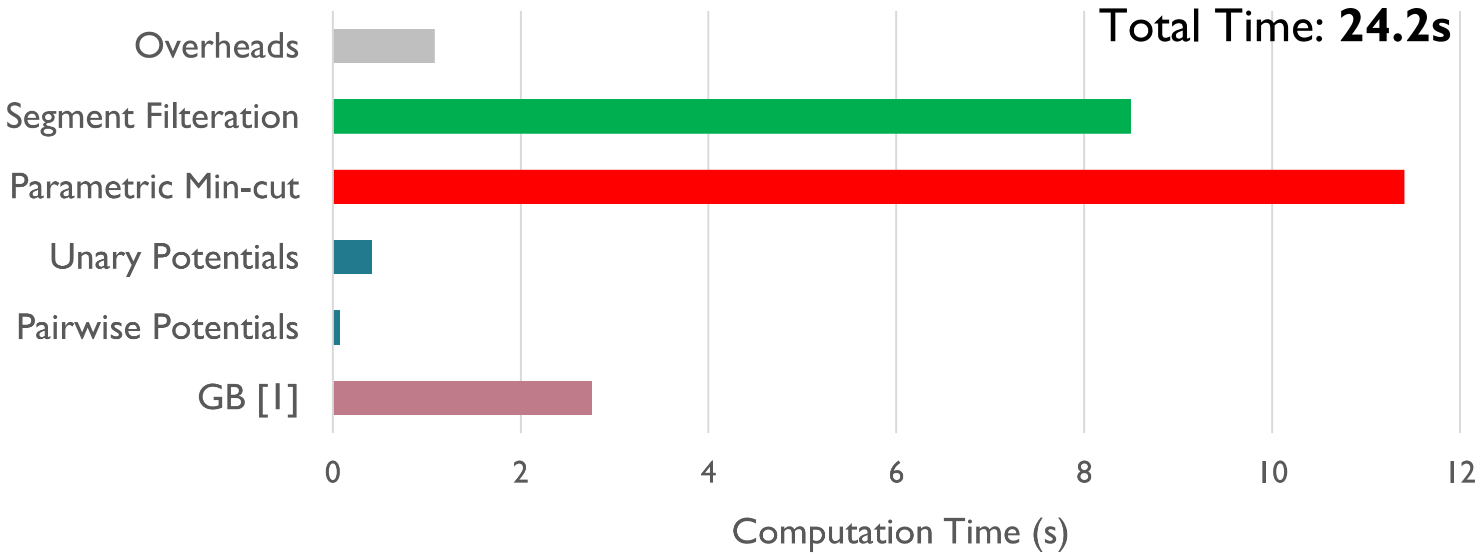
Seeds from Superpixels

[1] Leordeanu, *et al.*, "Efficient Closed-Form Solution to Generalized Boundary Detection," ECCV 2012
[2] Lim, Zitnick, and Dollar, "A Learned Mid-level Representation for Contour and Object Detection," CVPR 2013
[3] Dollar and Zitnick, "Structured Forests for Fast Edge Detection," ICCV 2013

Method Overview



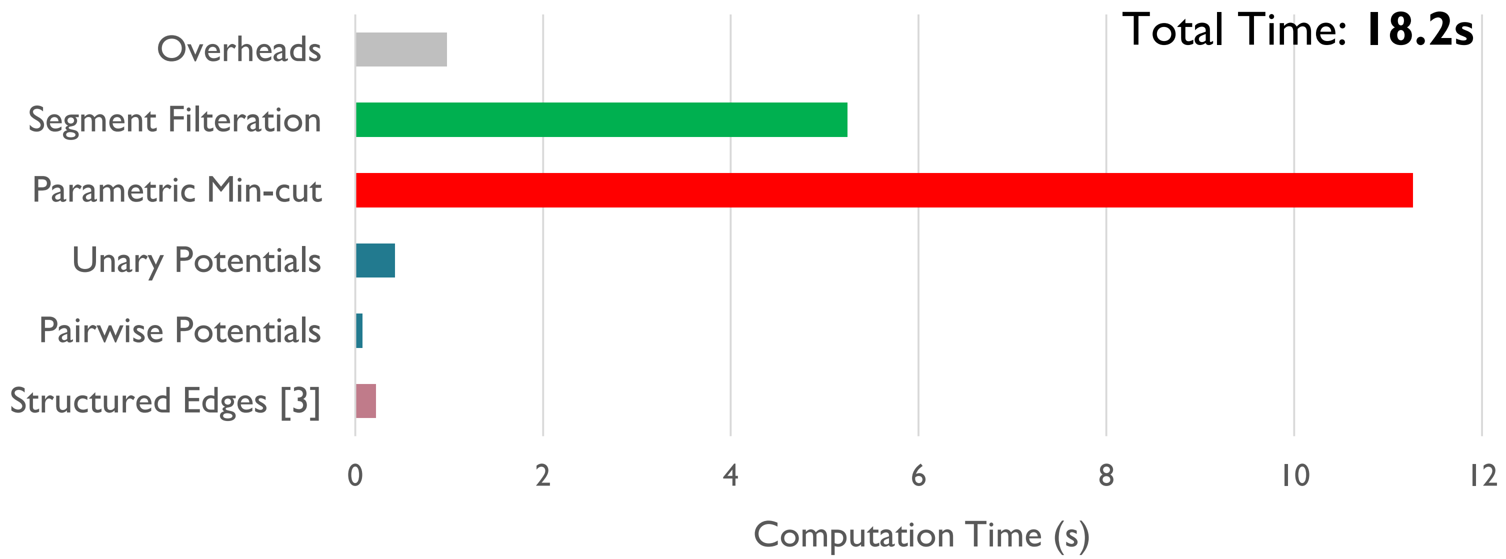
Computation Time



When using *Pixel* graphs (CPMC)

[1] Leordeanu, *et al.*, "Efficient Closed-Form Solution to Generalized Boundary Detection," ECCV 2012

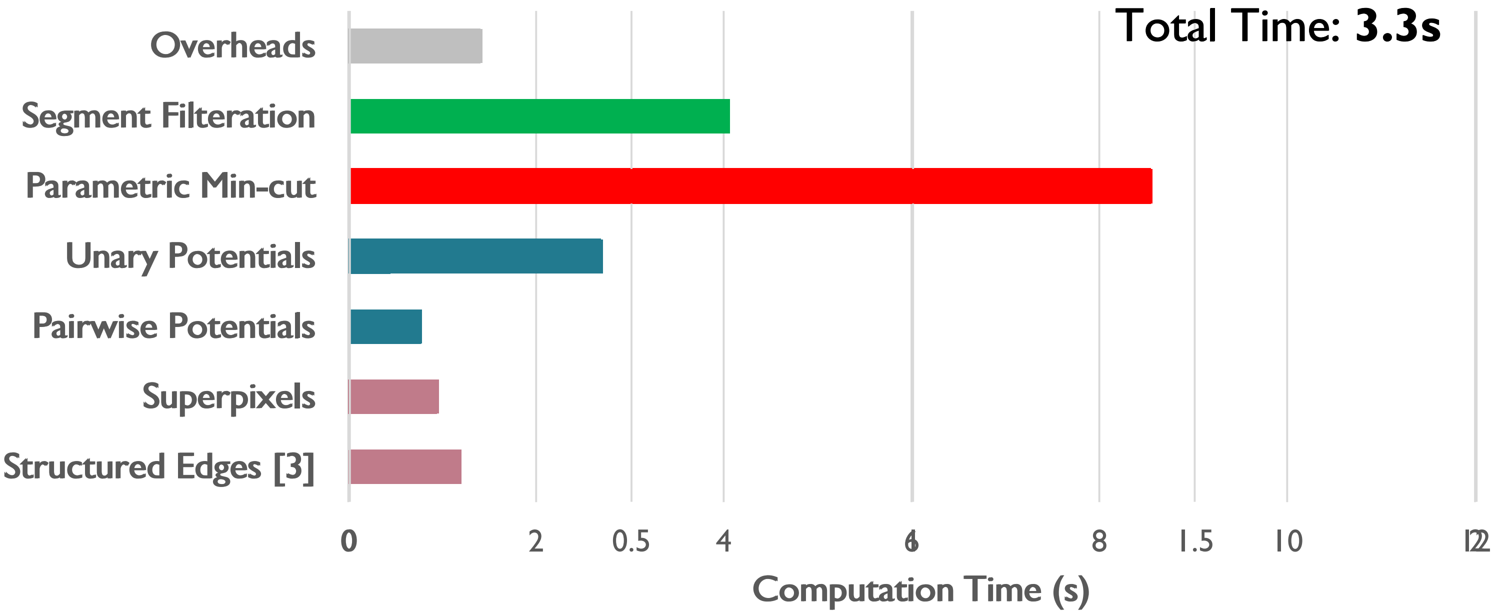
Computation Time



… use Structured Edges [3]

[3] Dollar and Zitnick, "Structured Forests for Fast Edge Detection," ICCV 2013

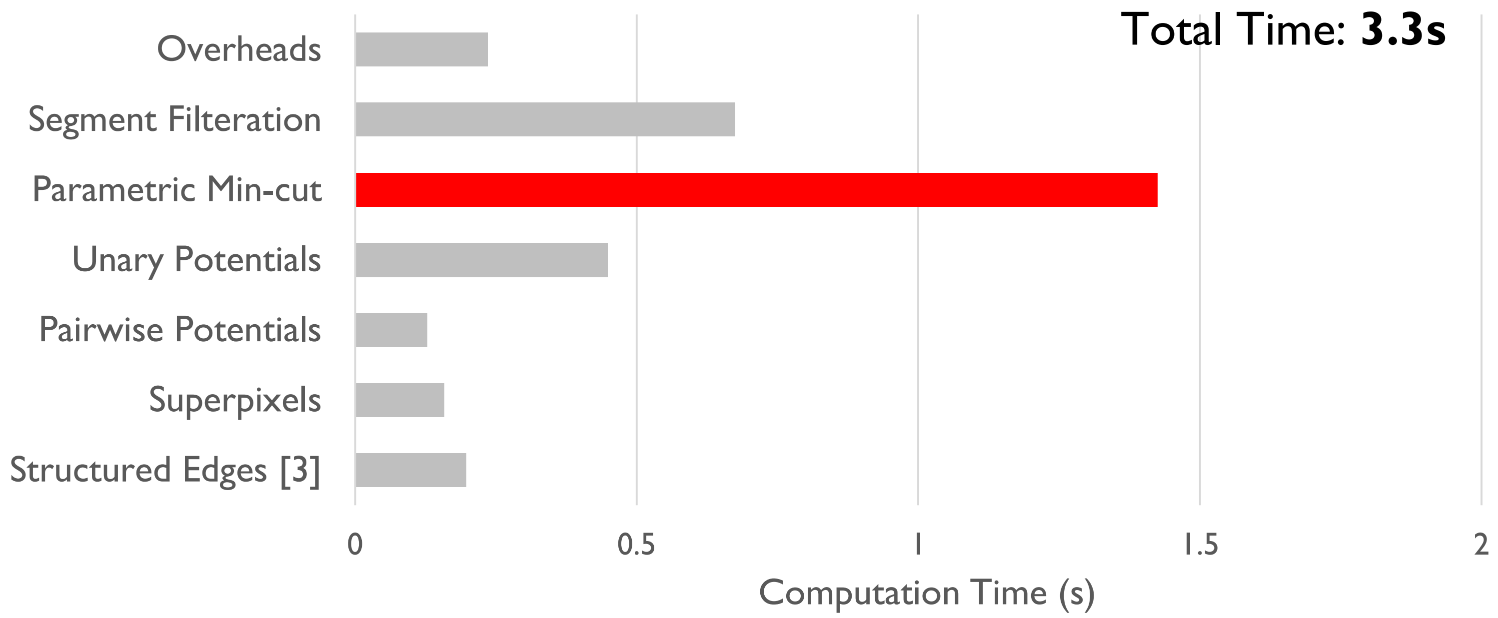
Computation Time



… woah … from *Pixels* to *Superpixels* based Graphs

[3] Dollar and Zitnick, "Structured Forests for Fast Edge Detection," ICCV 2013

Computation Time



How can we reduce the parametric Min-cut computation time?

[3] Dollar and Zitnick, "Structured Forests for Fast Edge Detection," ICCV 2013

Our Contributions

1. Method to reuse information for MAP inference in different. graphs with same pairwise costs, but different unary seeds.

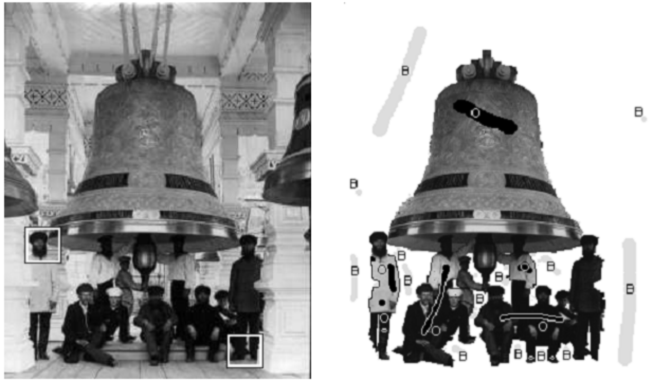
2. Allow sharing information across graph-cut problems before the full cut is computed (allows parallelization).

3. An object segmentation method which is an order of magnitude faster, without loss in accuracy.

Related Work

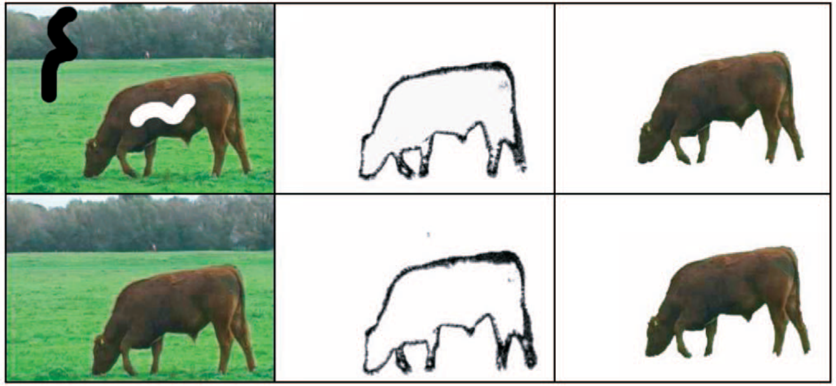
Boykov and Jolly [1]

Reusing Flows for Interactive Segmentation



Kohli and Torr [2]

Reusing Flows in Dynamic Graph-Cuts



[1] Boykov and Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images," ICCV 2001
[2] Kohli and Torr, "Dynamic Graph Cuts for Efficient Inference in Markov Random Fields," PAMI 2007

How to use Parametric Min-Cut?

Input Image, I

Labeling of all Superpixels

Pairwise Superpixel Potentials

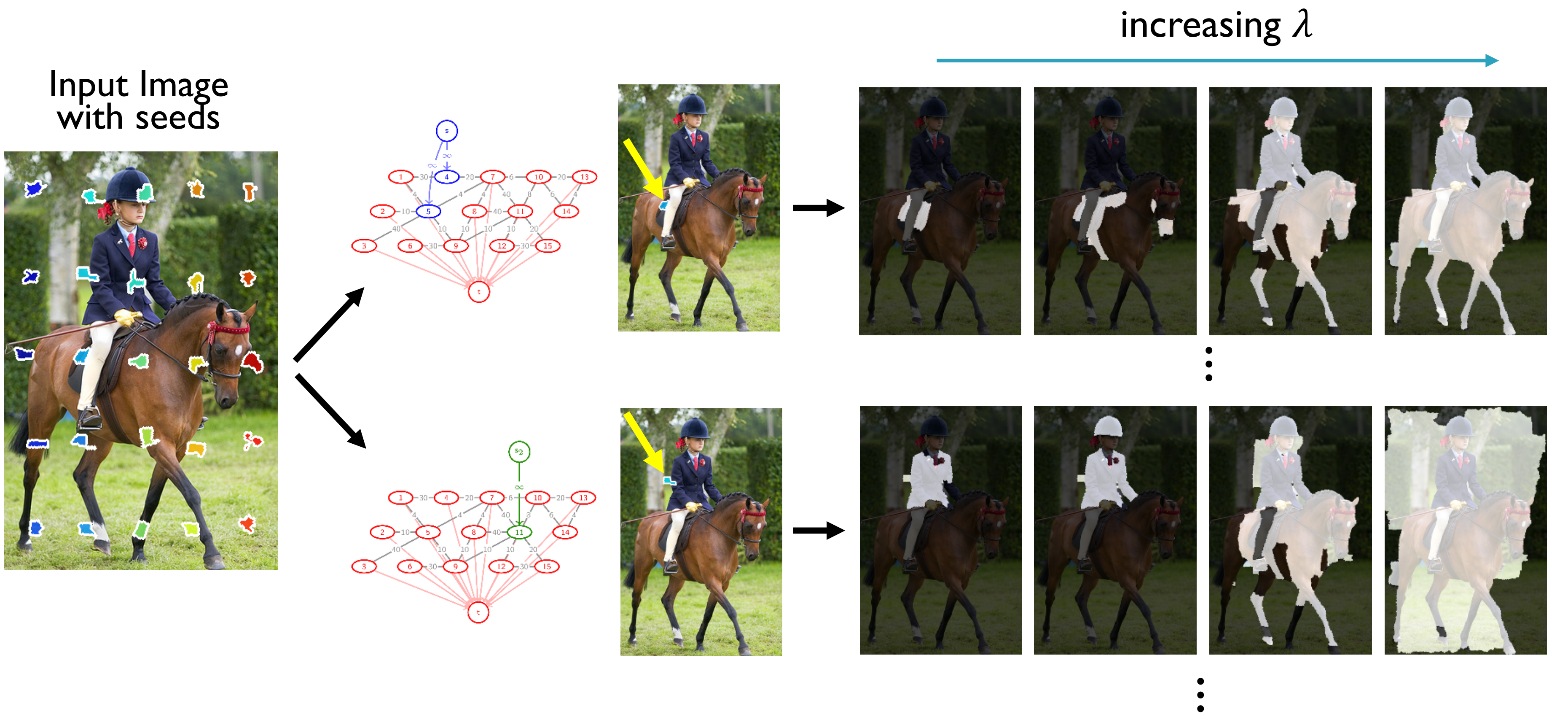
$$E_\lambda^i(\boldsymbol{X}) = \sum_{u\in\mathcal{V}} \underbrace{D_\lambda^i(x_u)}_{\text{Parametric Unaries}} + \sum_{(u,v)\in\mathcal{E}} \overbrace{V_{uv}(x_u, x_v)}^{\text{Pairwise Superpixel Potentials}}$$

Parametric Unaries

$D_\lambda^i(x_u) = \infty$ — If $u \in S_i$ and assigned to background

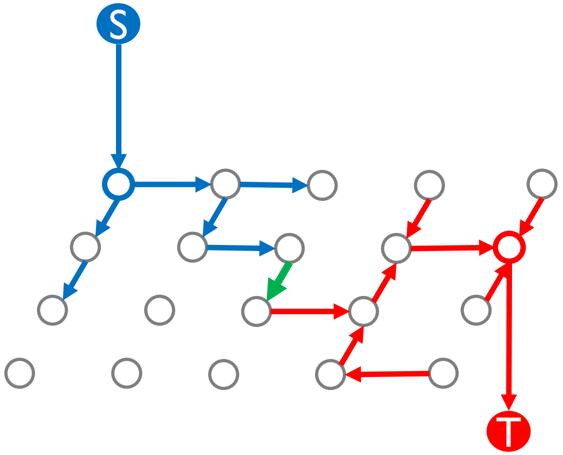
$D_\lambda^i(x_u) = f(x_u) + \lambda$ — Parametric unaries otherwise

How to use Parametric Min-Cut?



Preliminary: Boykov-Kolmogorov (BK)

Grow Trees



T sink tree edge

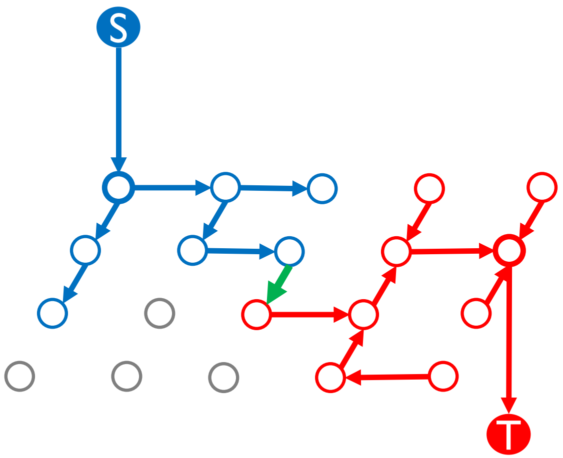
S sink tree edge

Augmenting path from *S* to *T*

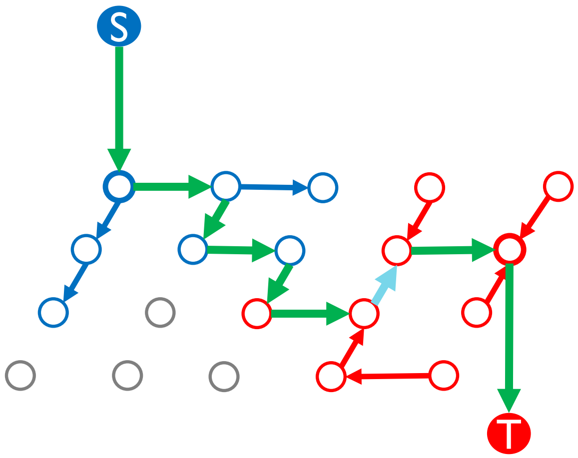
[1] Boykov and Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," PAMI 2004

Preliminary: Boykov-Kolmogorov (BK)

Grow Trees



Augment Flow



T sink tree edge

S sink tree edge

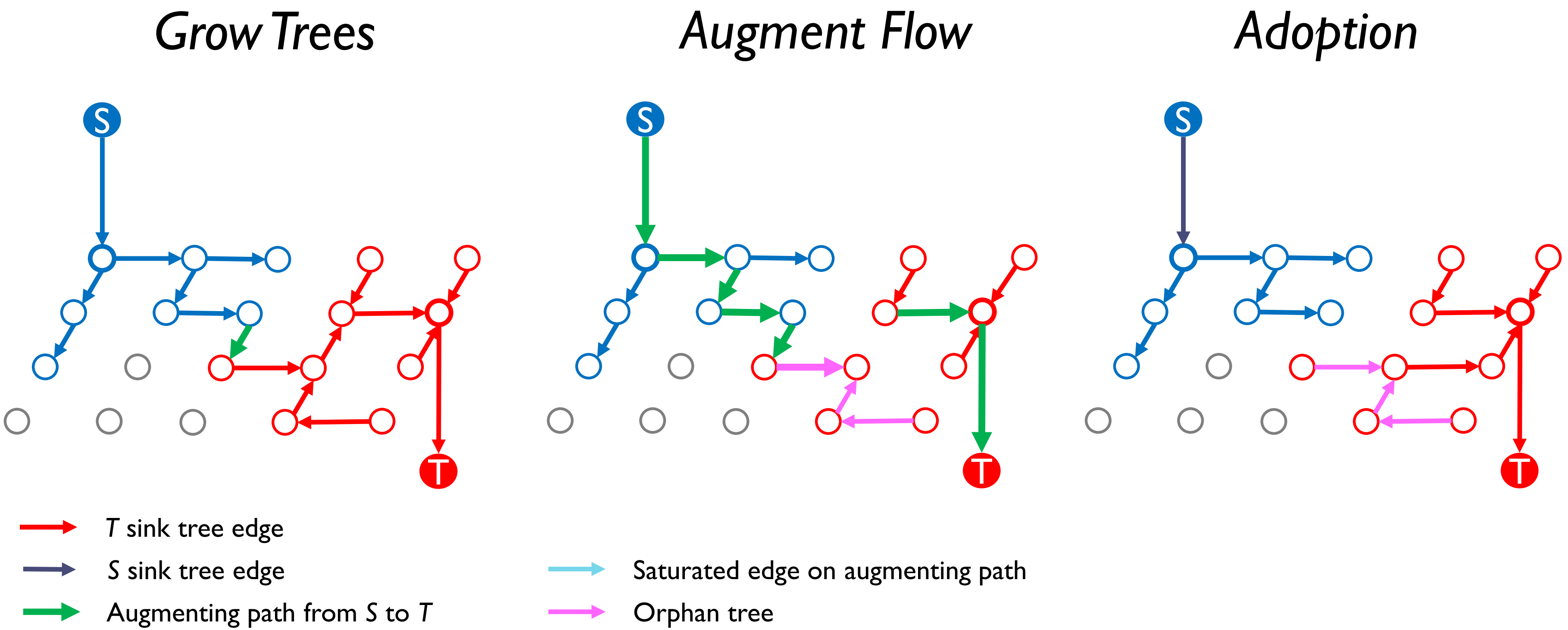
Augmenting path from *S* to *T*

Saturated edge on augmenting path

Orphan tree

[1] Boykov and Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," PAMI 2004

Preliminary: Boykov-Kolmogorov (BK)



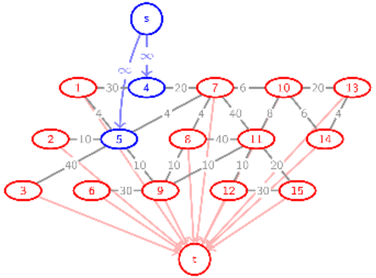
[1] Boykov and Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," PAMI 2004

Problem Statement

Given N energy functions for parametric min-cut, find what information can be shared for their minimization (MAP). V_{uv} remains same across the functions.

Seed: $D_\lambda^i(x_u) = \infty$ iff $x_u \in S_i$ and $x_u = 0$. **Condition**: $S_i \cap S_j = \emptyset$, for all i, j

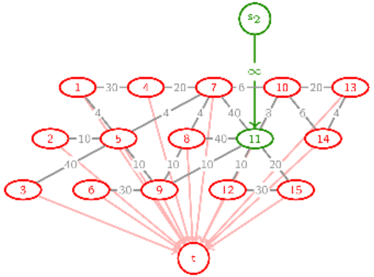
S_i



$$E_\lambda^i(\boldsymbol{X}) = \sum_{u \in \mathcal{V}} D_\lambda^i(x_u) + \sum_{(u,v) \in \mathcal{E}} V_{uv}(x_u, x_v)$$

Share information

S_j

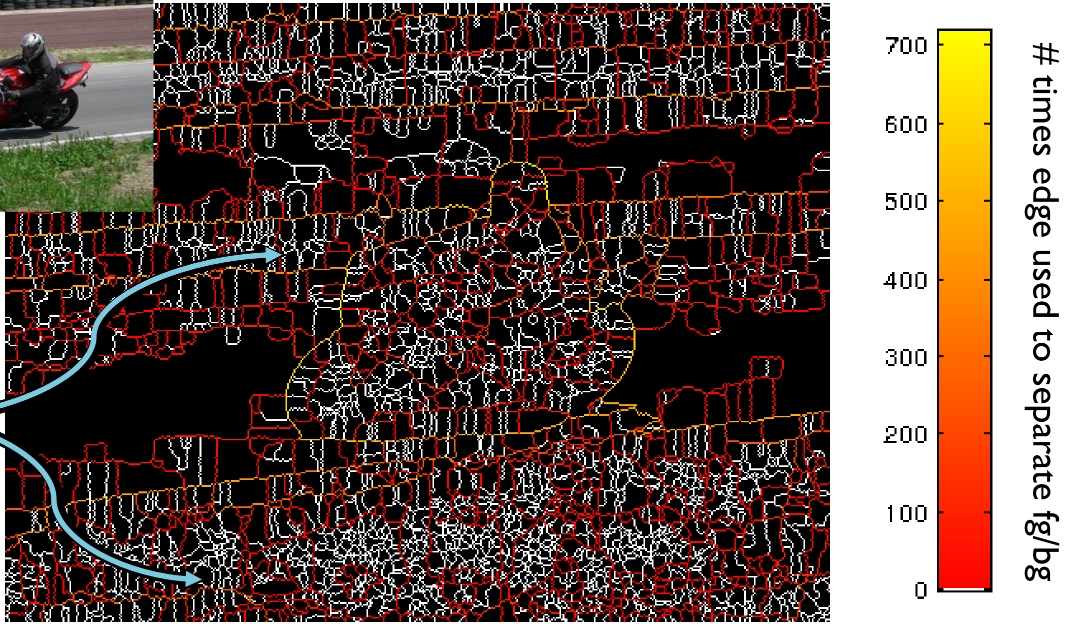


$$E_\lambda^j(\boldsymbol{X}) = \sum_{u \in \mathcal{V}} D_\lambda^j(x_u) + \sum_{(u,v) \in \mathcal{E}} V_{uv}(x_u, x_v)$$

Key Insight

Collect all parametric min-cut segments.

- Count how many times each superpixel edgelet was in the cut

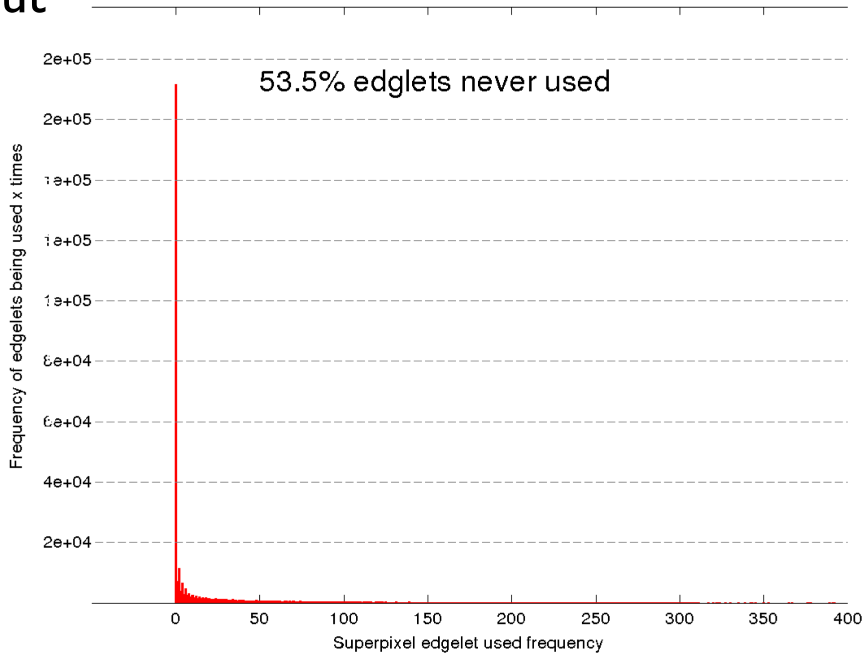


Lots of white edges, which are never used in a cut

Key Insight

~54% boundaries never in the cut

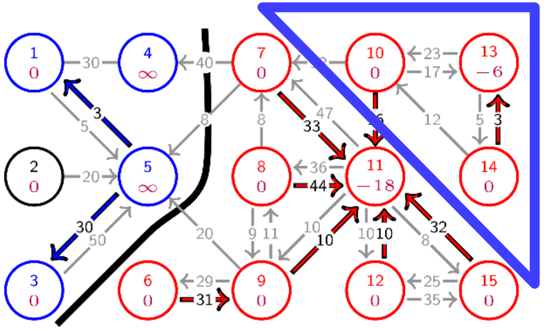
- **Share information across Graph-cuts** – communicate that some edglets never in cut



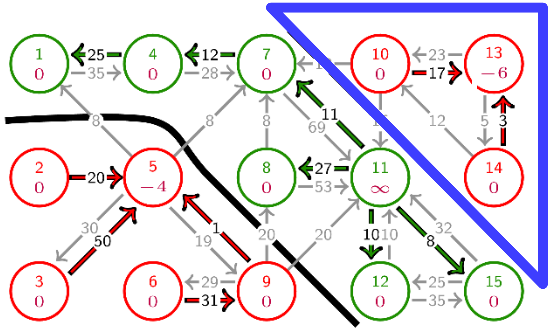
What can we Reuse?

BK spends time creating trees – reusing them is useful

Cut for seed S_1



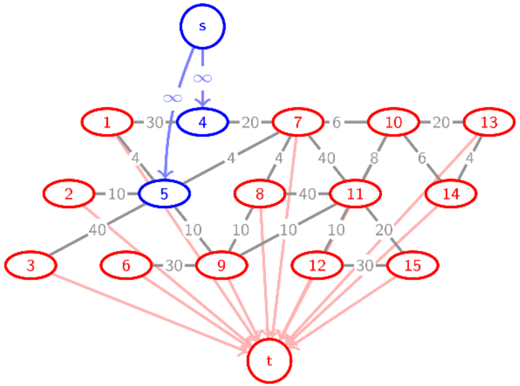
Cut for seed S_2



Idea - generate trees that are useful across all seeds

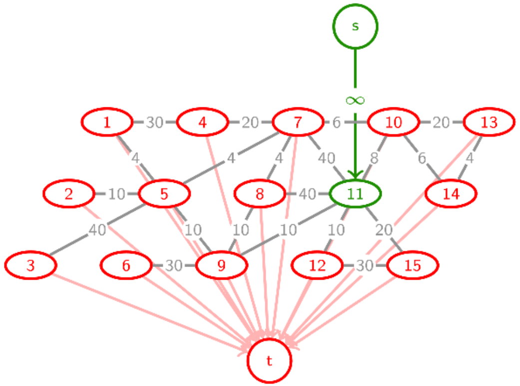
1st step: combine all seeds into one precomputation graph

Seed S_1 graph



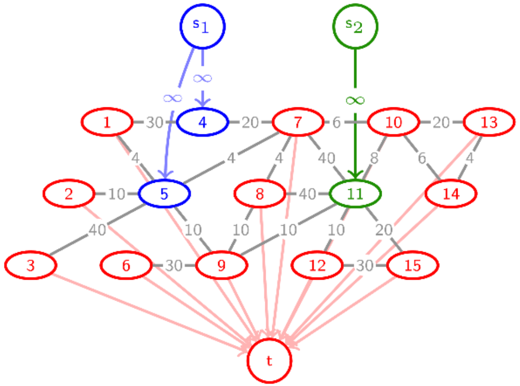
∪

Seed S_2 graph



=

Precomputation graph

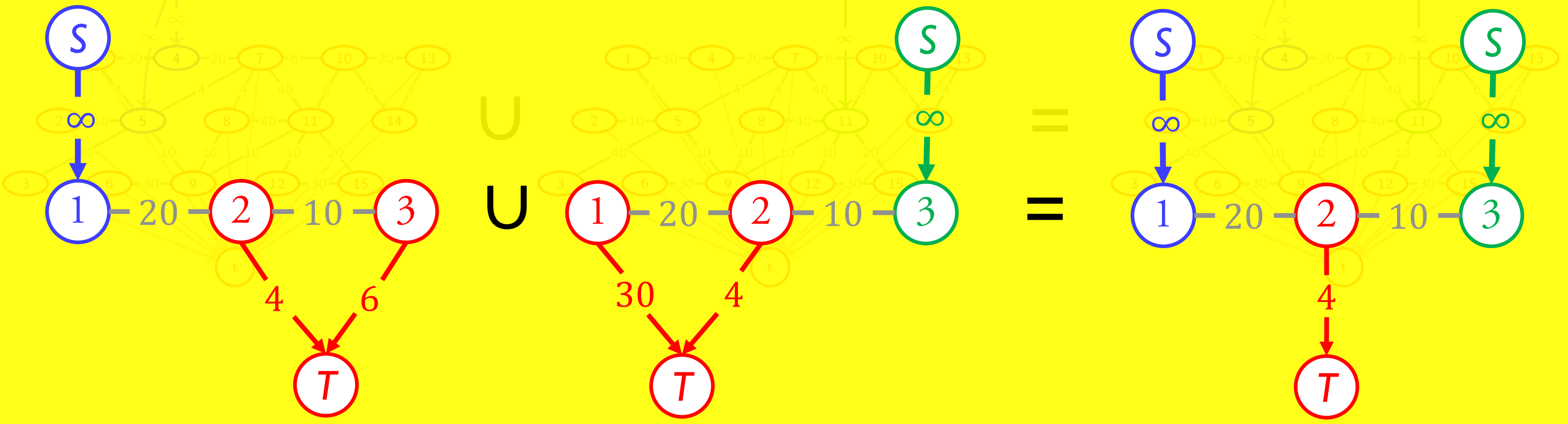


Idea - generate trees that are useful across all seeds

1st step: combine all seeds into one precomputation graph

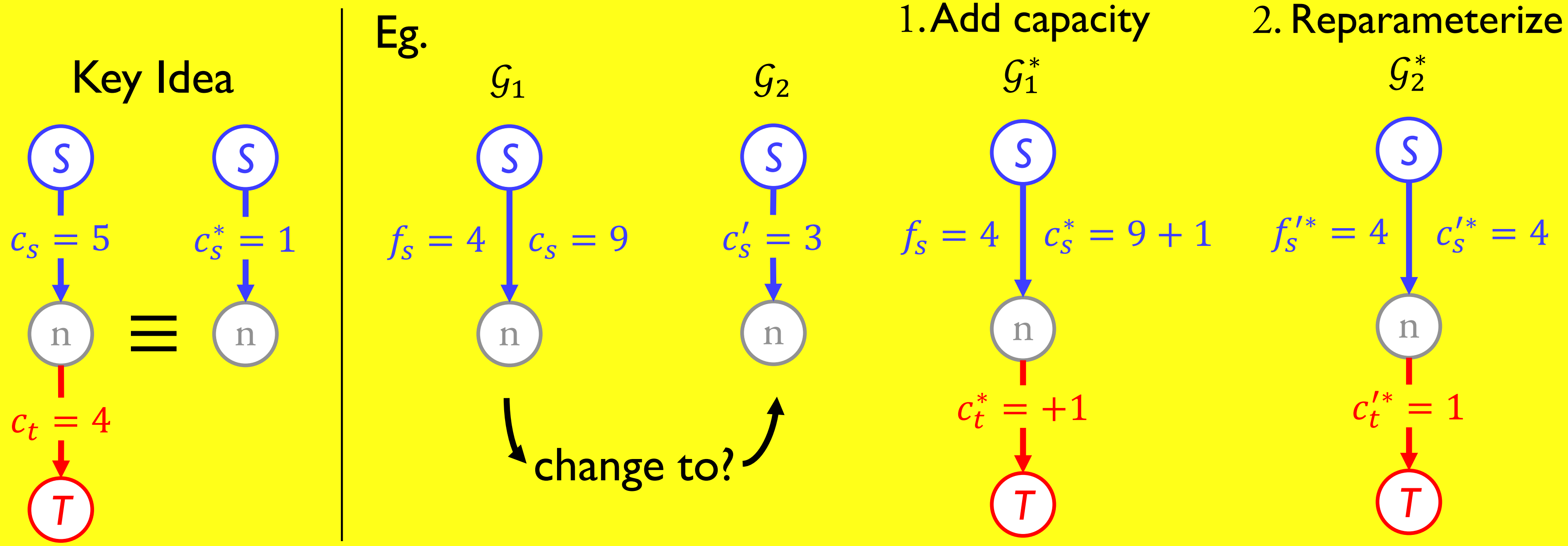
Example

Seed S_1 graph ∪ Seed S_2 graph = Precomputation graph



Reparameterization [1,2]

Changes flow value but not the cut! As long as $c_s - c_t$ remains same

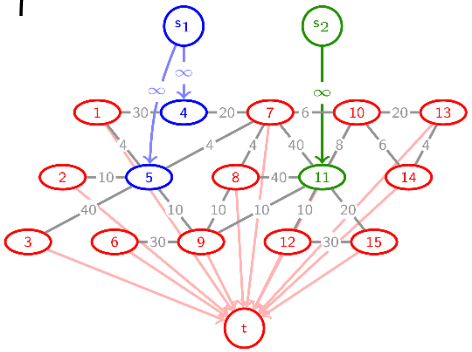


[1] Boykov and Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images," ICCV 2001
[2] Kohli and Torr, "Dynamic Graph Cuts for Efficient Inference in Markov Random Fields," PAMI 2007

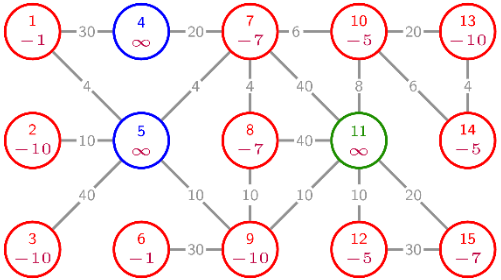
Reusing computation by Precomputation Graph

2^{nd} step: Run Boykov-Kolmogorov on precomputation graph

Precomputation graph

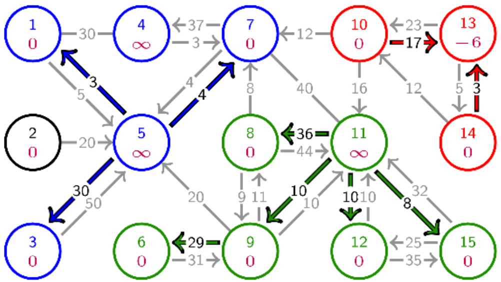


$\equiv$



Run B.K. for growing trees

Solved Precomputation graph

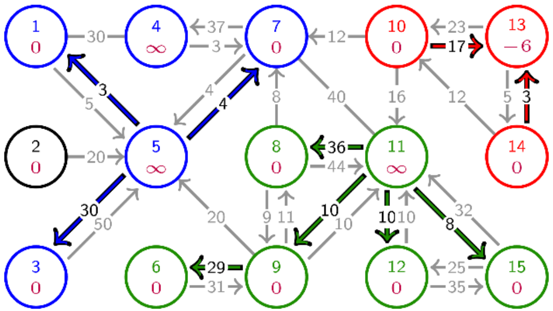


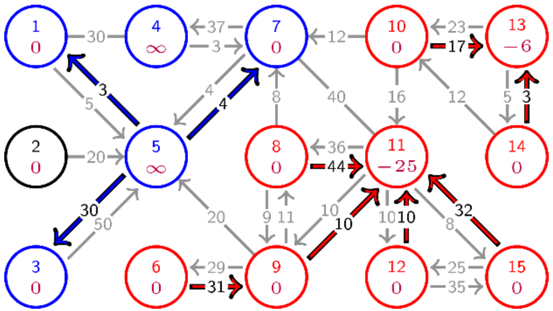
Notice, we now have two S trees and one T tree

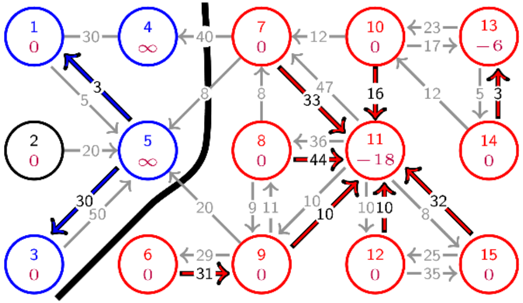
Reusing computation by Precomputation Graph

3[rd] step: Convert Precomputation Graph to compute max-flow for each seed

Solved Precomputation graph ⟶ Reparameterized to use for seed S_1 ⟶ Maxflow for seed S_1





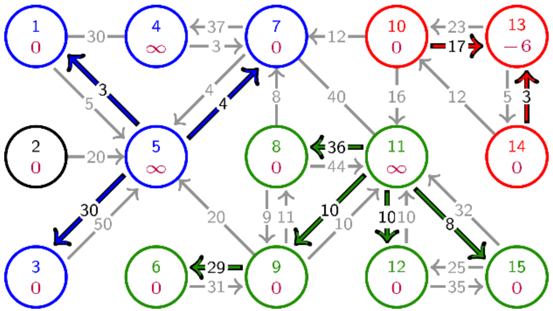


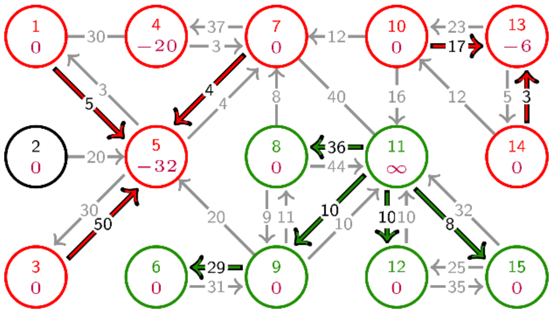
Note: we needed to convert tree S_2 to T

Reusing computation by Precomputation Graph

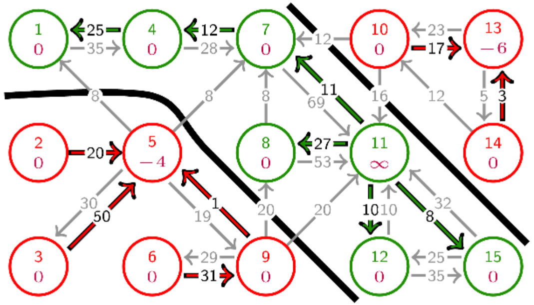
3^{rd} step: Repeat for all seeds

Solved Precomputation graph → Reparameterized to use for seed S_2 → Maxflow for seed S_2





Tree S_1 converted to T

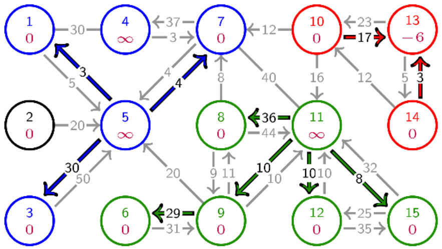


Note: *we get the same cuts, because we are just reparameterizing*

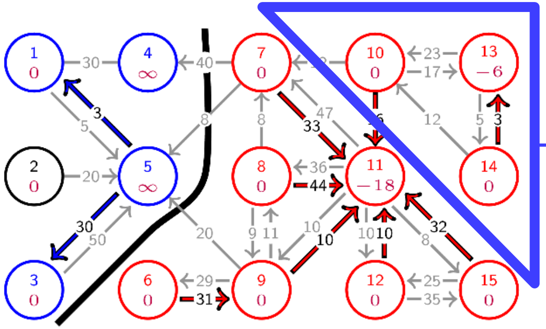
Why is the Precomputation Graph useful?

BK spends time creating trees – reusing them is useful

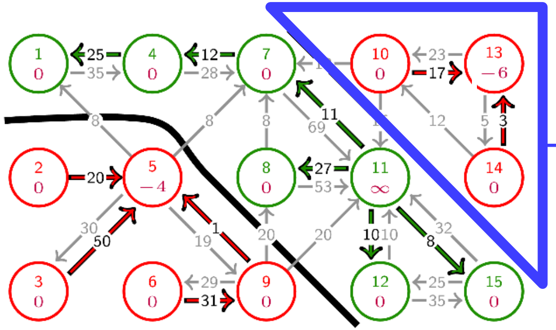
Solved Precomputation graph



Cut for seed S_1



Cut for seed S_2

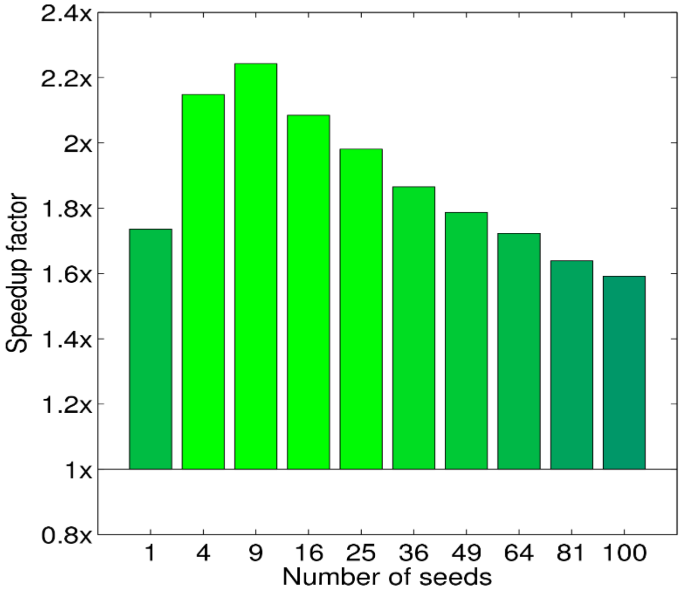


Completely reused from Precomputation Graph

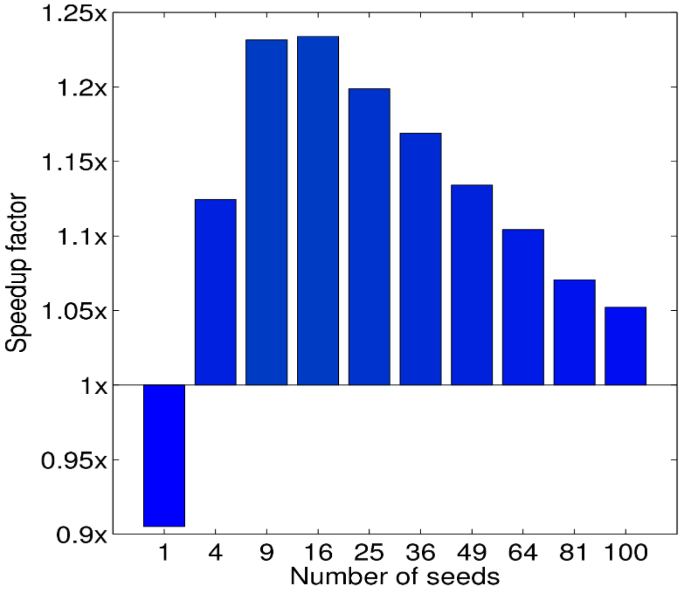
Speed-up with Precomputation Graph

Parametric Min Cut time savings

Compared to Boykov Kolmogorov [1]



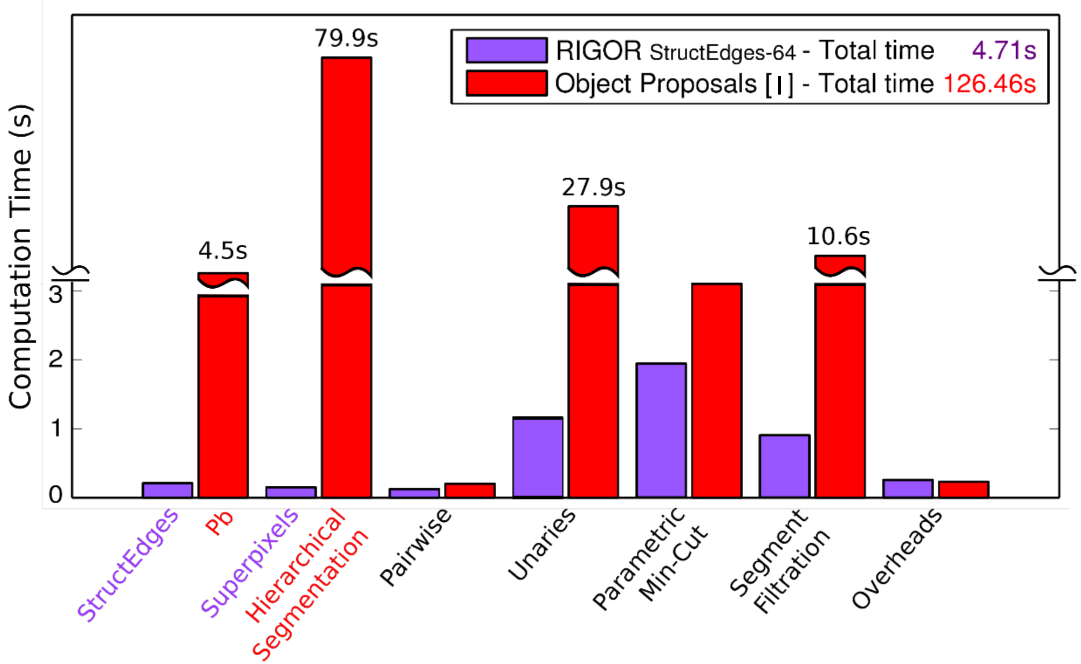
Compared to Kohli & Torr [2]



[1] Boykov and Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," PAMI 2004
[2] Kohli and Torr, "Dynamic Graph Cuts for Efficient Inference in Markov Random Fields," PAMI 2007

Faster Pipeline

Pipeline timing comparison to Object Proposals [1]



[1] Endres and Hoiem, "Category-Independent Object Proposals with Diverse Ranking," PAMI 2013

Quantitative Comparison

Method		Mean Best Overlap	Mean Best Covering	Run Time (s)	# of Segments
CPMC		70.67	82.24	34.01	624.1
Object Proposals		71.48	80.98	126.46	1544.1
Shape Sharing		67.82	82.71	410.31	1115.4
RIGOR	GB, 25 seeds	68.04	79.83	4.62	808.7
	StructEdges, 25 seeds	68.85	79.89	**2.16**	741.9
	GB, 64 seeds	72.83	82.55	6.99	1490.3
	StructEdges, 64 seeds	73.64	82.84	4.71	1462.8
	GB, 100 seeds	74.22	83.25	9.26	1781.9
	StructEdges, 100 seeds	**75.19**	**83.52**	6.84	1828.7

Future Directions

1. Learning unaries which are faster to compute, and accurate.
2. Parametric min-cut for both unaries and pairwise energies. [1]
3. Multiple precomputation graphs, each only dealing with graph with similar unary costs.
4. GPU implementation.

[1] Lim, Jung, and Kohli, "Efficient Energy Minimization for Enforcing Statistics," PAMI 2004

Summary

1. Presented a method to precompute some information for different graph-cuts.
2. Our method can find re-use parts before computing full cuts.
3. A practical object segmenter, running under 2 secs!
4. CODE : http://cpl.cc.gatech.edu/projects/RIGOR