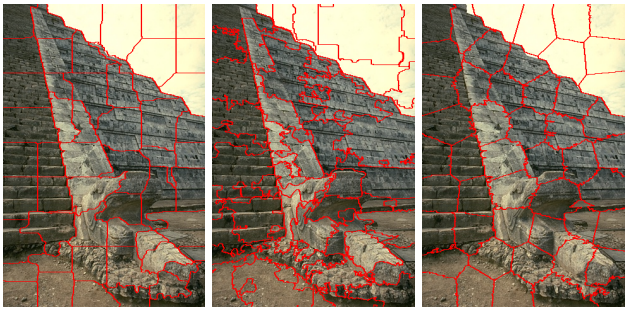
Real-Time Coarse-to-fine Topologically Preserving Segmentation

Jian Yao[1], Marko Boben[2], Sanja Fidler[1], Raquel Urtasun[1]

[1]University of Toronto, [2]University of Ljubljana

Abstract

In this paper, we tackle the problem of unsupervised segmentation in the form of superpixels. Our main emphasis is on speed and accuracy. We build on [31] to define the problem as a boundary and topology preserving Markov random field. We propose a coarse to fine optimization technique that speeds up inference in terms of the number of updates by an order of magnitude. Our approach is shown to outperform [31] while employing a single iteration. We evaluate and compare our approach to state-of-the-art superpixel algorithms on the BSD and KITTI benchmarks. Our approach significantly outperforms the baselines in the segmentation metrics and achieves the lowest error on the stereo task.



Ours SEEDS [7] SLIC [1]

Figure 1. Our approach generates regular superpixels that preserve boundary and topology. Notice how our approach better snaps to image boundaries.

1. Introduction

Superpixels have become an established image preprocessing step to significantly reduce the complexity of higher-level computer vision techniques. They have been applied in a wide variety of domains such as image labeling [32, 25], object proposal generation [6], optical flow estimation [31], tracking [2], and stereo [28]. Typically their role is to partition the image into a tractable set of "units", which make the consequent processing run several order of magnitude faster while at the same time do not sacrifice their performance. Superpixels also serve as an efficient regularizer for domains such as segmentation, stereo or flow estimation, where pixel-level estimates are noisy or even missing.

A useful superpixel algorithm should thus ideally run fast, possibly real-time, and guarantee a reliable, regular, and topologically coherent image partitioning. While a majority of existing work does not satisfy most of these requirements, particularly speed [5, 21, 33, 18], some of the recent work reported real-time running times [1, 8]. One of the fastest approaches is SLIC [1] which performs iterative k-means style clustering, but which does not ensure that the final segmentation is connected. A post-processing step is typically needed to correct for this issue, however, this step is likely to increase the overall labeling energy. This was addressed by [31] who proposed an optimization strategy that only considers updating the boundary pixels at each iteration. Their segmentation method additionally incorporates length of the boundary as shape regularization, as well as evidence from stereo image pairs. The final super pixel algorithm then reasons jointly about the partitioning as well as the stereo estimation in the form of slanted planes.

In this paper, we build on [31] and propose a much more efficient optimization algorithm that results in an order of magnitude less updates (speed-up). Inspired by the SEEDS algorithm [8] our method uses a coarse-to-fine energy update strategy, which allows the optimization to reach better energy minima than [31] when employing even a single iteration.

We demonstrate the effectiveness of our approach in two important settings: unsupervised segmentation of RGB images, as well as joint segmentation and stereo estimation via slanted planes. Our results on BSD [19] demonstrate that our approach significantly outperforms SLIC in all segmentation metrics and SEEDS under the boundary recall metric. Furthermore, we outperform [31] in a single iteration, achieving a much better minima of the labeling energy. We test the stereo super pixels on the autonomous driving dataset KITTI [11], where real-time processing is of partic-

ular importance. We show a significant improvement over the SLIC and SEEDS baselines, and an order of magnitude speed-up over [31] in terms of the number of boundary updates needed for convergence. Importantly our approach outperforms the state-of-the-art when taking occluded and non-occluded pixels into account, demonstrating the power of our slanted plane super pixel algorithm.

2. Related Work

A wide variety of superpixel methods have been proposed in the last few years [1, 8, 17, 26, 12, 20]. The main differences are in the objective function they minimize and in the optimization technique that performs the minimization. These are typically based on agglomerative clustering in the color domain [9, 12, 14], k-means style energy optimization [1], and coarse-to-fine optimization [7, 8]. While speed is typically one of the most important requirements, only a few algorithms actually run in real-time [1]. Time however is a crucial factor, especially for application domains such as robotics and autonomous driving where the little processing time should be spent on inferring the high order semantics and not on low level processing. Our main focus here is on real-time running times.

Most superpixel algorithms partition the image into homogeneous color regions with some regularization constraints that encourage the resulting regions to be smooth and regular, yet well preserve the image boundaries [1, 31]. In order to preserve the region topology, superpixel lattices have been used [20], or clever energy updates where only the boundary pixels are considered at each step [31, 7]. Our work follows this line of work. We build on the work of Yamaguchi et al. [31], and show how a coarse-to-fine optimization inspired by [8] can result in an order of magnitude speed-up. With respect to [8], we optimize a very different energy function, which among other terms differ in the regularization term, which [8] due to the nature of their parametrization cannot handle efficiently. In our approach regularization plays an important role as demonstrated by our experimental evaluation.

In recent years slanted plane methods that jointly reason about stereo/flow and super pixels have been proposed [28, 29, 27]. While they perform very well in challenging scenarios, with the exception of [31] they are however computationally very expensive, limiting their applicability to real-wold applications such as autonomous driving. Here we build on [31] and show that by using a coarse to fine strategy, an order of magnitude speed up can be achieved at the same time as better stereo estimates.

3. Efficient Topology Preserving Segmentation

We start our discussion by describing our segmentation algorithm when only a monocular image is available. In the next section we describe our slanted plane segmentation algorithm, when we have access to a stereo pair.

Algorithm 1 Coarse to Fine Monocular Segmentation

```
Initialize the superpixel to be a regular grid;
Compute initial μ̂_i and ĉ_i for each segment i;
for l = 1 to levelMax do
  Initialize each block on level l with a regular grid;
  Compute mean color and position in each block;
  for iter = 1 to maxIters do
    Initialize list with all boundary blocks on level l;
    while list is not empty do
      pop out boundary block b_i^l from list;
      if valid_connectivity(b_i^l) then
        ŝ_{b_i^l} = argmin_{s_{b_i^l}} E_mono(s, μ̂, ĉ)
        if s_{b_i^l} is updated then
          compute μ̂ and ĉ incrementally for the two
          superpixels involved;
          append neighbors of b_i^l if they are boundary
          to the list end;
        end if
      end if
    end while
  end for
end for
```

3.1. Monocular Super-pixel Estimation

Let $s_p \in \{1, \cdots, M\}$ be the assignment of pixel p to a superpixel, and let $\mathbf{s} = (s_1, \cdots, s_N)$ be the set of all random variables representing the segmentation, with N the size of the image. Following [31], we formulate the segmentation problem with an objective function similar to k-means clustering, where we want superpixels that are coherent in appearance but that have also regular shape. We additionally add constraints on the size of the superpixel to prevent tiny superpixels.

Let μ_i be the mean position of the i-th superpixel and let c_i be its mean color. Our Markov random field (MRF) energy is then defined as

$$E_{mono}(\mathbf{s}, \boldsymbol{\mu}, \mathbf{c}) = \sum_p E_{col}(s_p, c_{s_p}) + \lambda_{pos} \sum_p E_{pos}(s_p, \mu_{s_p}) + \lambda_b \sum_p \sum_{q \in \mathcal{N}_8} E_b(s_p, s_q) + E_{topo}(\mathbf{s}) + E_{size}(\mathbf{s}) \quad (1)$$

with $\mathbf{c} = (c_1, \cdots, c_M)$, $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_M)$ the set of centers and mean positions for all superpixels, and $\mathcal{N}_8$ the 8 neighborhood of pixel p. We now define the energy terms in more detail.

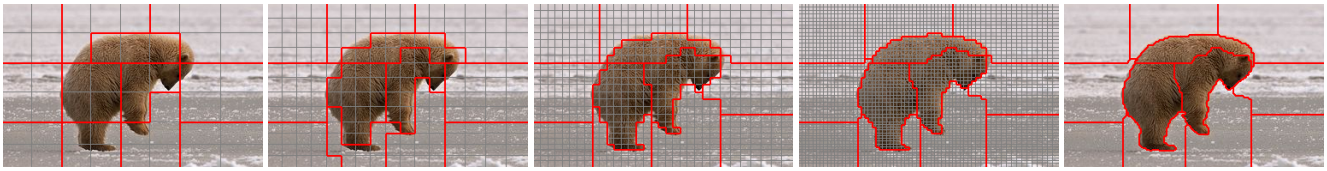


Figure 2. Coarse-to-fine boundary-level updates start at the coarse level (left) and proceeds to the finest level iteratively. The final result, defined on the finest (pixel) level, is shown on the right.

Algorithm 2 Coarse to Fine Stereo Segmentation

Initialize superpixels with a regular grid;
Compute initial $\hat{\mu}_i$ and $\hat{c}_i$ for each segment i;
Compute the θ_i using RANSAC;
for $l = 1$ to levelMax **do**
  Initialize each block on level l with a regular grid;
  Compute $\hat{\mu}$ and $\hat{c}$ in each block;
  for $iter = 1$ to maxIters **do**
    Initialize list with all boundary blocks on level l;
    while list is **not empty do**
      pop out boundary block b_i^l from list;
      if valid_connectivity(b_i^l) **then**
        $\{\hat{s}_{b_i^l}, \hat{f}_{b_i^l}\} = \text{argmin}\, E_{total}(\mathbf{s}, \hat{\boldsymbol{\mu}}, \hat{\mathbf{c}}, \hat{\boldsymbol{\theta}}, \hat{\mathbf{o}})$
        if $\hat{s}_{b_i^l}$ is updated **then**
          compute $\hat{\mu}$ and $\hat{c}$ incrementally for the two superpixels involved;
          append neighbors of b_i^l if they are boundary to the list end;
        end if
      end if
    end while
    for k = 1 to *numIters* **do**
      $\hat{\mathbf{o}} = \text{argmin}_{\mathbf{o}}\, E_{total}(\hat{\mathbf{s}}, \hat{\boldsymbol{\mu}}, \hat{\mathbf{c}}, \hat{\boldsymbol{\theta}}, \mathbf{o})$
      $\hat{\boldsymbol{\theta}} = \text{argmin}_{\theta_i}\, E_{total}(\hat{\mathbf{s}}, \hat{\boldsymbol{\mu}}, \hat{\mathbf{c}}, \boldsymbol{\theta}, \hat{\mathbf{o}})$
    end for
  end for
end for

Shape Regularization: This term imposes that the superpixels should be regular in shape

$$E_{pos}(s_p, \mu_{s_p}) = ||p - \mu_{s_p}||_2^2 \tag{2}$$

where μ_{s_p} is the superpixel centroid.

Appearance Coherence: This term encourages color homogeneity of each superpixel:

$$E_{col}(s_p, c_{s_p}) = (\mathcal{I}(p) - c_{s_p})^2 \tag{3}$$

where c_{s_p} is the mean color descriptor for superpixel s_p.

Boundary Length: We further impose regularization by encouraging the superpixels to have small boundary length

$$E_b(s_p, s_q) = \begin{cases} 1 & \text{if } s_p \neq s_q \\ 0 & \text{o.w.} \end{cases} \tag{4}$$

Topology Preservation: This term forces the superpixels to form a connected component and penalizes ∞ otherwise.

Minimum size: We force the superpixels to be at least $1/4$ size of their initialization. This potential has value ∞ if this constraint is not satisfied.

3.2. Coarse-to-Fine Optimization

SLIC superpixels [1] minimize the sum of appearance homogeneity and shape regularization using coordinate descent, where the algorithm iterates between estimating the assignments $\mathbf{s}$ for all pixels, and computing the mean position and color $\boldsymbol{\mu}$, $\mathbf{c}$ for all superpixels. Estimating $\boldsymbol{\mu}$ and $\mathbf{c}$ can be done in closed form, by computing the empirical mean of the positions and colors of the pixels belonging to each superpixel. The superpixel assignments are computed by iterating over each pixel in a sequential fashion. The latter process is not very efficient, as the algorithm iterates across all pixels even though at each iteration only a subset change assignment. Furthermore, the result of this algorithm is not topologically correct (i.e., superpixels might contain holes as well as discontinuities). This can be enforced by encoding it in terms of the boundary pixels, but the optimization of which is very difficult to solve when employing a k-means style optimization.

In [31], the optimization was done by maintaining a queue, which is initialized with the pixels at the boundary. The pixels in the queue are then iteratively popped out and discarded if minimizing the energy does not change their current assignment. If the assignment changes, the new boundary pixels are pushed to the top of the priority queue. As a consequence, if there is no boundary length energy, [31] typically gets stuck and runs in vein.

In this paper, we propose both a coarse-to-fine optimization algorithm for the pixel assignment as well as a FIFO strategy for the priority queue. As shown in our experimental evaluation this will allow us to estimate segmentation in a fraction of the time required by SLIC [1] or [31].

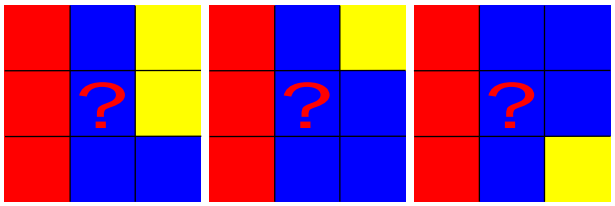


Figure 3. Forbidden cases: Changing the middle block from blue to red violates connectivity.

Our algorithm starts by initializing the superpixels to a regular grid, and computing the initial estimates for the mean position and color for each superpixel. We then iteratively optimize every level (from coarse to fine). This allows us to make fast moves (at coarser levels) reaching a much better local optimum of our objective function. During the computation at each level, we first initialize each block to a regular grid and compute mean color and position for each block. We initialize the priority list with all blocks that form a boundary. We then take a boundary block from the list, and check if changing the label will violate connectivity (see Fig. 3). If it does not violate, we solve optimally for the block assignment by minimizing the energy in Eq. (1). This is done by simply trying all assignments from the 4-neighboring blocks. If the block has changed assignment, we update the mean position and color using the incremental mean equation for the two superpixels involved (the one that this block belonged to before, as well as the one in the new assignment). Given the previous estimate m_{n-1} and a new element a_n, the mean can be computed incrementally

$$m_n = m_{n-1} + \frac{a_n - m_{n-1}}{n}$$

with n the size of the k-th superpixel after the update. Finally, we push the neighbors of the block if they are at the boundary at the bottom of the priority queue (having lowest priority). We repeat this process until the list is empty, at which point we switch to the new level. Note that at the finest level a block is the same as a pixel, and at coarser level a block is a set of pixels. We summarize our algorithm in Algo 1 and refer the reader to Fig. 2 for an illustration of the results for each level (from coarse-to-fine).

4. Efficient Joint Segmentation and Stereo

In this section we tackle the problem of joint segmentation and stereo estimation when a stereo pair is available.

4.1. Holistic Energy

Following slanted-plane methods [31, 28, 4], we represent the disparity of a superpixel with a slanted plane $\theta_i = (A_i, B_i, C_i)$ and reason about segmentation in the left image. The disparity of a pixel belonging to the i-th superpixel can then be computed by

$$d(\mathbf{p}, \theta_i) = A_i p_x + B_i p_y + C_i, \tag{5}$$

We further define $o_{i,j} \in \{co, hi, lo, ro\}$ to be a discrete variable that reasons about the type of occlusion boundary between adjacent superpixels i and j, with the states representing whether the boundary is co-planar, hinge, the i-th plane is in front, or behind. Let $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_M)$ be the set of plane parameters for all superpixels and let $\mathbf{o}$ be the set of all occlusion variables. Additionally, let f_i be an outlier flag for the i-th pixel, and let $\mathbf{f}$ the set of flags for all pixels. We define the energy of the joint segmentation and stereo estimation as the sum of energies encoding the monocular energy as well as consistency with the stereo image evidence, a prior on the complexity of the boundaries and smoothness between slanted planes of neighboring super pixels. Thus

$$E_{total}(\mathbf{s}, \boldsymbol{\mu}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{o}, \mathbf{f}) = E_{mono}(\mathbf{s}, \boldsymbol{\mu}, \mathbf{c}) + E_{stereo}(\mathbf{s}, \boldsymbol{\theta}, \mathbf{o}, \mathbf{f}) \tag{6}$$

with the energy related to stereo defined as

$$\begin{aligned} E_{stereo}(\mathbf{s}, \boldsymbol{\theta}, \mathbf{o}, \mathbf{f}) = \lambda_{disp} &\sum_{p \in \mathcal{F}} E_{disp}(s_p, \theta_{s_p}, f_p) \\ &+ \lambda_{smo} \sum_{(i,j) \in \mathcal{N}_{seg}} E_{smo}(\mathbf{s}, \theta_i, \theta_j, o_{i,j}) \\ &+ \lambda_{prior} \sum_{(i,j) \in \mathcal{N}_{seg}} E_{prior}(o_{i,j}) \end{aligned} \tag{7}$$

with $\mathcal{N}_{seg}$ the set of superpixels that form a boundary, and $\mathcal{F}$ the set of pixels where SGM returns an estimate. We now describe the stereo energy terms in more details.

Disparity Coherence: This term encodes the fact that the disparity of the slanted plane should be consistent with the image evidence. Towards this goal, we make use of semi-global block matching (SGM) [15] to provide an initial estimate. This allow us to speed-up the slanted plane formulation inference to be real time. Note that real-time implementations of SGM exist in both CPU and GPU [3, 24]. In particular, we use a truncated quadratic function which makes the estimation of disparity plane parameters more robust to outliers.

$$E_{disp}(s_p, \theta_{s_p}, f_p) = \begin{cases} d(p, \theta_{s_p}) - \hat{d}(p))^2 & \text{if } f_p = 0 \\ \lambda_d & \text{otherwise} \end{cases} \tag{8}$$

with β a scalar, and $\hat{d}(p)$ the SGM disparity estimate.

Occams Razor: This term encodes a prior over the complexity of the model, which prefers simpler explanations,

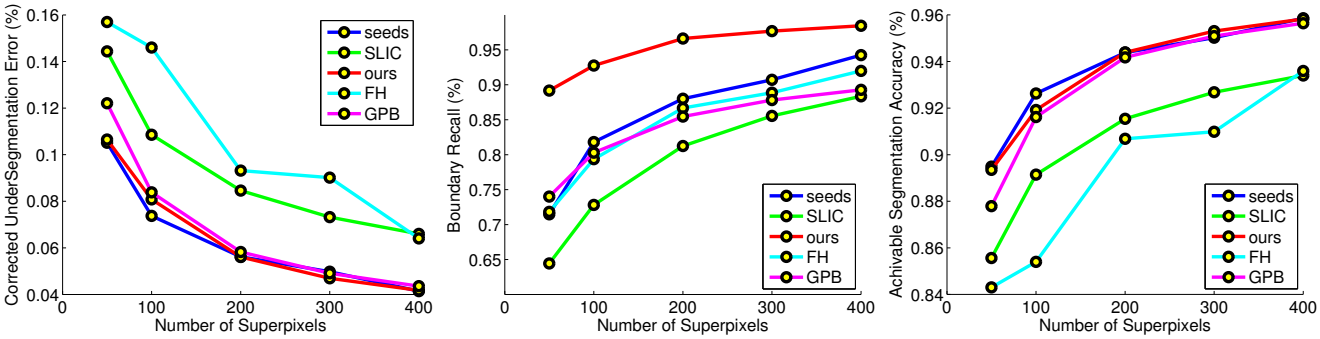


Figure 4. **BSD:** In this experiment we use the position weight to be 0 for both our approach and SLIC to mimic SEEDS that does not use regularization. Note that our approach outperforms both baselines. Speed: SEEDS(30Hz), SLIC(5HZ), OURS(30HZ)

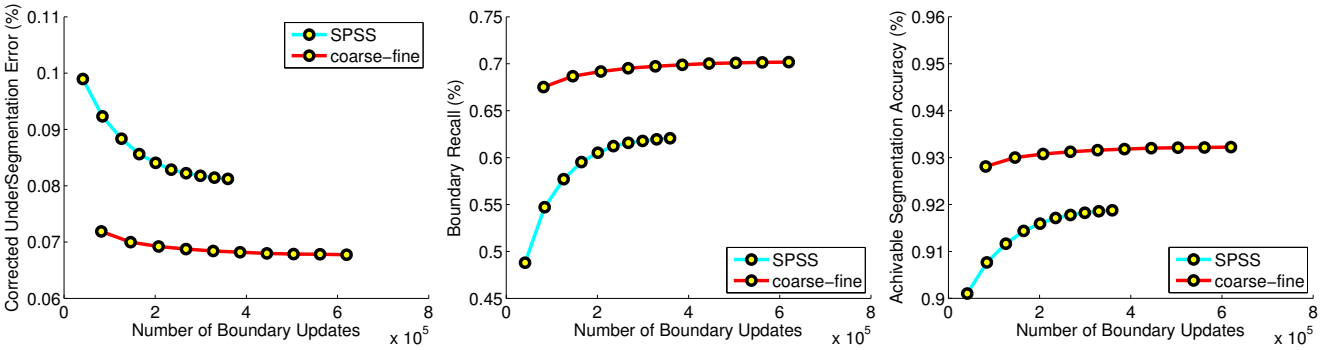


Figure 5. **BSD**: Comparison to [31] as a function of the number of boundary updates.

i.e., coplanar over hinge, hinge over occlusion.

$$E_{prior}(o_{i,j}) = \begin{cases} \lambda_{\text{occ}} & \text{if } o_{i,j} = lo \ \vee \ o_{i,j} = ro \\ \lambda_{\text{hinge}} & \text{if } o_{i,j} = hi \\ 0 & \text{if } o_{i,j} = co \end{cases} \tag{9}$$

where $\lambda_{\text{occ}}, \lambda_{\text{hinge}}$ are constants and $\lambda_{\text{occ}} > \lambda_{\text{hinge}} > 0$.

Spatial Smoothing: The smoothness term imposes the planes of adjacent superpixels to have similar disparity in the shared boundary if they form a hinge and all over the superpixel if they are co-planar.

$$E_{smo}(\mathbf{s}, \theta_i, \theta_j, o_{i,j}) = \begin{cases} \phi_{occ}(o_{i,j}) & \text{if } o_{i,j} = occ \\ q(\theta_i, \theta_j, \mathbf{B}_{i,j}) & \text{if } o_{i,j} = hi \\ q(\theta_i, \theta_j, \mathbf{S}_i \cup \mathbf{S}_j) & \text{if } o_{i,j} = co \end{cases} \tag{10}$$

with $\mathbf{B}_{i,j}$ the set of pixels in the boundary of the two superpixels, $\mathbf{S}_i \cup \mathbf{S}_j$ the pixels in the union of the two superpixels, and

$$q(\theta_i, \theta_j, \Omega) = \frac{1}{|\Omega|} \sum_{p \in \Omega} (d(p, \theta_i) - d(p, \theta_j))^2$$

Note that $\phi_{occ}(o_{i,j})$ forces the labeling of $o_{i,j}$ to pick the plane in front if there is an occlusion.

4.2. Coarse-to-Fine Optimization

Similarly to the monocular case, we can optimize jointly over segmentation and stereo variables using block coordinate descent. This algorithm iterates between solving for the plane parameters, the occlusion variables, the color and position means as well as the assignments. Similarly to the monocular case, we extend [31] to have a FIFO priority queue as well as a coarse-to-fine strategy to minimize the objective. As shown in our experimental evaluation, this reduces the computation time by an order of magnitude while achieving better performance.

Let $\mathbf{b}^l$ be the set of blocks at level l. Note that at the finer level, the number of blocks is the number of pixels. Our coarse to fine algorithm proceeds as follows: we first initialize the superpixels to a grid and compute the initial estimates of mean color and position for each segment. We then use RANSAC to initialize the plane parameters for each superpixel. The algorithm then iterates for each level, from the coarser to the finer. For each level, we initialize the blocks to a regular grid, the mean positions and colors to their respective empirical estimates as well as the boundary list to contain all blocks which are in the boundary. We then follow the monocular case, and re-estimate the local variables using incremental updates whenever a boundary is changed. When the boundary list is empty, the algorithm computes

the occlusion variables in a sequential manner. The boundary variables are optimized one at a time by selecting the label with minimum cost. The outlier flags are optimized given the current plane parameters. The planes are then fitted using least squares taking into account the non-outlier pixels only. Our coarse-to-fine algorithm is summarized in Algo 2. As shown in our experimental evaluation, both by using a FIFO priority queue and a coarse to fine strategy, our approach speeds up inference significantly over other efficient slanted plane algorithms, e.g., [31].

5. Experiments

To evaluate our approach in the monocular setting, we use the Berkeley Segmentation Dataset (BSD500) [19], which provides multiple ground truth contours and segmentations. All our results are evaluated on the test set which contains 200 images. For the stereo setting, we use the KITTI stereo dataset [11] which has 194 pairs of images in training and 195 in test. The ground truth is semi-dense and has been captured using a Velodyne LIDAR.

We use standard metrics to evaluate the performance for both unsupervised segmentation and stereo. For unsupervised segmentation, we use: Corrected Undersegmentation Error (CUE), Boundary Recall (BR) and Achievable Segmentation Accuracy (ASA). Let $\mathcal{S} = \{s_1, \cdots, s_m\}$ be our segmentation and let the ground truth be $\mathcal{G} = \{g_1, \cdots, g_n\}$. We define the *Corrected Under Segmentation* as

$$CUE(\mathcal{S}, \mathcal{G}) = \frac{\sum_j |s_j - \text{argmax}_{g_i \in \mathcal{G}} |s_j \cap g_i||}{\sum_i |g_i|}$$

Our second metric is the *Boundary Recall* defined as:

$$BR(\mathcal{S}, \mathcal{G}) = \frac{\sum_i [\min_j |B^j_{\mathcal{S}} - B^i_{\mathcal{G}}| < \epsilon]}{|B_{\mathcal{G}}|}$$

where $B_{\mathcal{G}}$ and $B_{\mathcal{S}}$ are the set of boundary pixels from the ground truth and the segmentation respectively. The superscript indicates the element in the boundary set, and ϵ is a constant. Finally, the *achievable segmentation accuracy* is defined as:

$$ASA(\mathcal{S}, \mathcal{G}) = \frac{\sum_j \max_i |s_j \cap g_i|}{\sum_i |g_i|}$$

To measure the accuracy of the stereo estimation we use the percentage of erroneous pixels which disparity estimate is farther than a fixed number of pixels. We do so both in non-occlusion areas (Out-NOC) as well as in all pixels (Out-All). We additionally report average disparity error in non-occluded areas (Avg-Noc) and in all pixels (Avg-All). These are the standard metrics in the KITTI dataset.

Comparison on BSD500: We first compare our method to SLIC, SEEDS as well as the methods of [21, 10]. Fig 4 shows performance as a function of the number of superpixels. SEEDS is also a coarse-to-fine algorithm and thus is a good baseline. Note that each plot corresponds to a different metric. Our method significantly outperforms SLIC under all metrics. We outperform SEEDS significantly in the Boundary Recall metric, and perform on-par with SEEDS for Corr. Undersegmentation Error and Achievable Segmentation Accuracy, although our approach is more stable. We show a few qualitative results in Fig. 6. One can notice that our superpixels better snap to the image boundaries. We also compare to [31], denoted as SPSS, in Fig. 5. The curves show different metrics as a function of the number of boundary updates. We can see that already in the first iteration our algorithm has almost converged, and there is little change in further iterations. SPSS ranks much lower than our approach in the first iteration, and remains significantly below our curve even after many iterations. This shows that our optimization technique is able to achieve a lower energy much faster than [31], and attains a better minimum overall. Note that our algorithm, similar to SEEDS runs at 30Hzs in this setup, while SLIC is only 5Hzs.

Comparison to the state-of-the-art on KITTI: We report the test errors for our model and competing approaches in Table 1. Our result uses 1000 superpixels and has been run for a single iteration. Note that our reported time, 1.7 seconds per image is dominated by the computation of SGM. However, there exists real-time GPU and CPU implementations of SGM which will make our full approach real-time. Our approach is also agnostic to which stereo algorithm takes as input, so other real time alternatives are also possible. In that case, our joint segmentation and slanted plane estimation will run in a fraction of [31]. Note also that our approach is the best performing one for all **All** settings, and on pair in the **NOC** settings. We also compare our approach SLIC and SEEDS in Fig. 7. Since SLIC and SEEDS do not tackle joint segmentation and stereo like us, we simply ran both methods on a single (left) RGB image, and robustly fit a slanted plane to their resulting superpixels using RANSAC. Both methods perform significantly lower than our approach.

Importance of the number of Superpixels: We compare our method to the baseline [31] in Fig 8, and a function of the number of superpixels per image. We run both methods for a single iteration. One can see that in the lower-superpixel regime our method significantly outperforms the baseline. The difference is smaller for larger number of super pixels (1000), yet still persist.

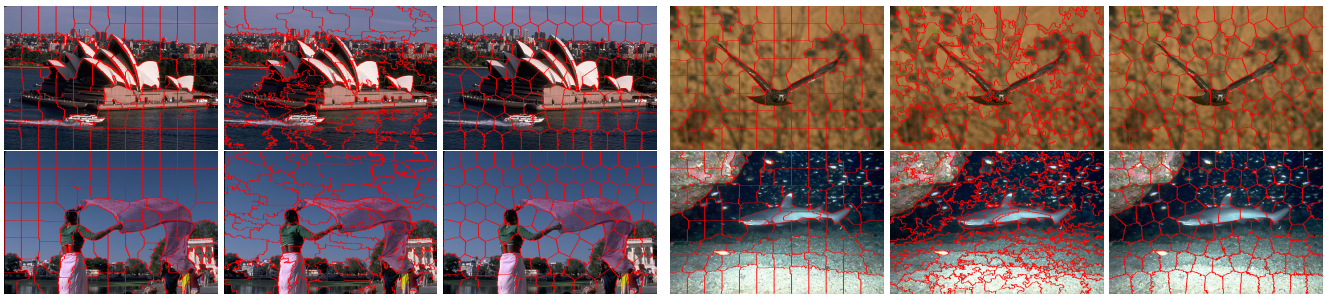


Figure 6. **BSD:** Qualitative results. Left to Right: Ours, SEEDS [7], SLIC [1]. Our approach better snaps to the image boundaries.

	> 2 pixels		> 3 pixels		> 4 pixels		> 5 pixels		End-Point		time
	NOC	**All**	**NOC**	**All**	**NOC**	**All**	**NOC**	**All**	**NOC**	**All**	
ALTGV [16]	7.88 %	9.30 %	5.36 %	6.49 %	4.17 %	5.07 %	3.42 %	4.17 %	1.1 px	1.2 px	20s
iSGM [13]	7.94 %	10.00 %	5.11 %	7.15 %	3.84 %	5.82 %	3.13 %	5.02 %	1.2 px	2.1 px	8s
ATGV [22]	7.08 %	9.05 %	5.02 %	6.88 %	3.99 %	5.76 %	3.33 %	5.01 %	1.0 px	1.6 px	360s
wSGM [23]	7.27 %	8.72 %	4.97 %	6.18 %	3.88 %	4.89 %	3.25 %	4.11 %	1.3 px	1.6 px	6s
PCBP [28]	6.08 %	7.62 %	4.04 %	5.37 %	3.14 %	4.29 %	2.64 %	3.64 %	0.9 px	1.1 px	300s
StereoSLIC [30]	5.76 %	7.20 %	3.92 %	5.11 %	3.04 %	4.04 %	2.49 %	3.33 %	0.9 px	1.0 px	2.3s
PCBP-SS [28]	5.19 %	6.75 %	3.40 %	4.72 %	2.62 %	3.75 %	**2.18** %	3.15 %	0.8 px	1.0 px	300s
SPS-St [31]	**4.98** %	6.28 %	**3.39** %	4.41 %	**2.72** %	3.52 %	2.33 %	3.00 %	0.9 px	1.0 px	2s
Ours	5.06 %	**5.09** %	3.41 %	**4.09** %	**2.72** %	**3.26** %	2.32 %	**2.79** %	0.9 px	1.0 px	1.7s

Table 1. **KITTI Stereo:** Comparison with the state-of-the-art on the test set of KITTI. We report test errors for our model with 1000 superpixels run for a single iteration. Note that our time is totally dominated by the computation of SGM. We achieve the best result for all **All** settings, that is, when taking all pixels including the occluding ones into account.

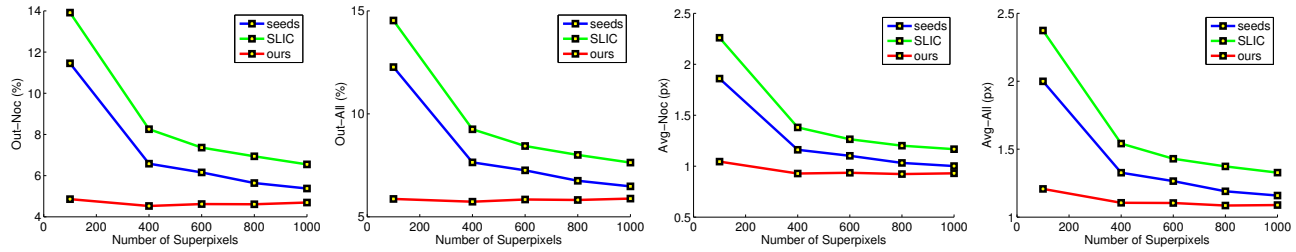


Figure 7. **KITTI stereo results**: Comparison to using SLIC and SEEDS to do segmentation follow by fitting a slanted plane via RANSAC. Results are reported for varying number of super pixels.

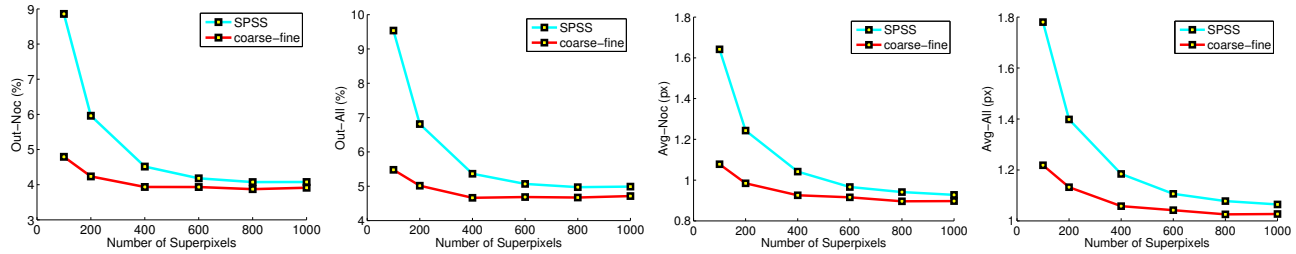


Figure 8. **Performance as a function of number of superpixels on KITTI:** comparison with [31] after running both algorithms for 1 iteration.

Importance of different features: We evaluate the effect of different features in the model. As shown in Fig. 9, a steady improvement in error exist every time a feature is added, and the best result is achieved when using all potentials. This justifies all the potentials used in our model.

Energy Optimization: As shown in Fig. 10, our coarse-to-fine approach converges very fast, as the updates are not local. faster than pixel level approach.

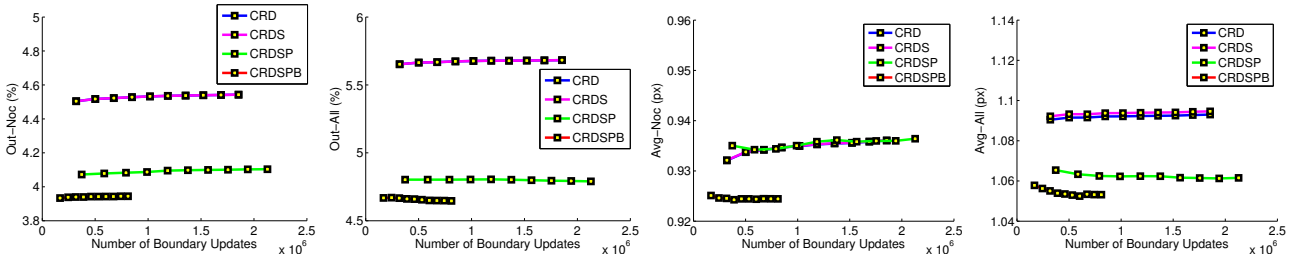
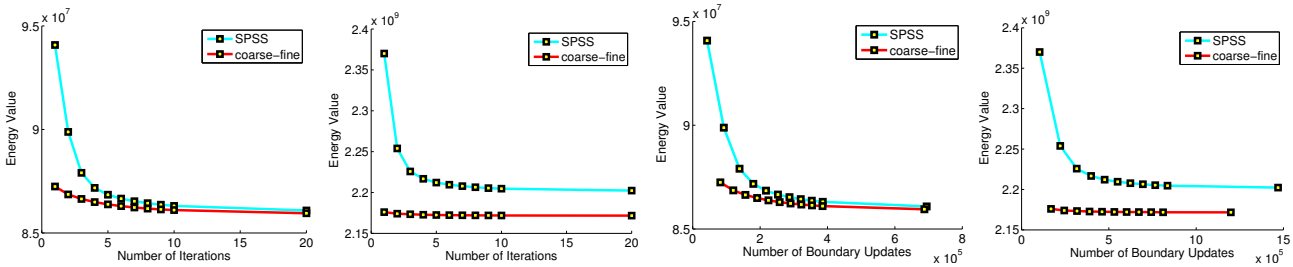


Figure 9. **KITTI: Importance of the features:** Evaluating the effect of different features on our results. All plots are for 400 superpixels. Different curves represent different combinations of the energy terms. We denote, **C**: color, **R**: Position (Shape Regularization), **D**: Disparity, **S**: Smoothing, **P**: Prior, and **B**: Boundary Length. Note that CRD and CRDS overlap in almost all plots.



(BSD) (KITTI) (BSD) (KITTI)

Figure 10. Energy decrease as a function of outer loop iterations and number of boundary updates in both BSD and KITTI.

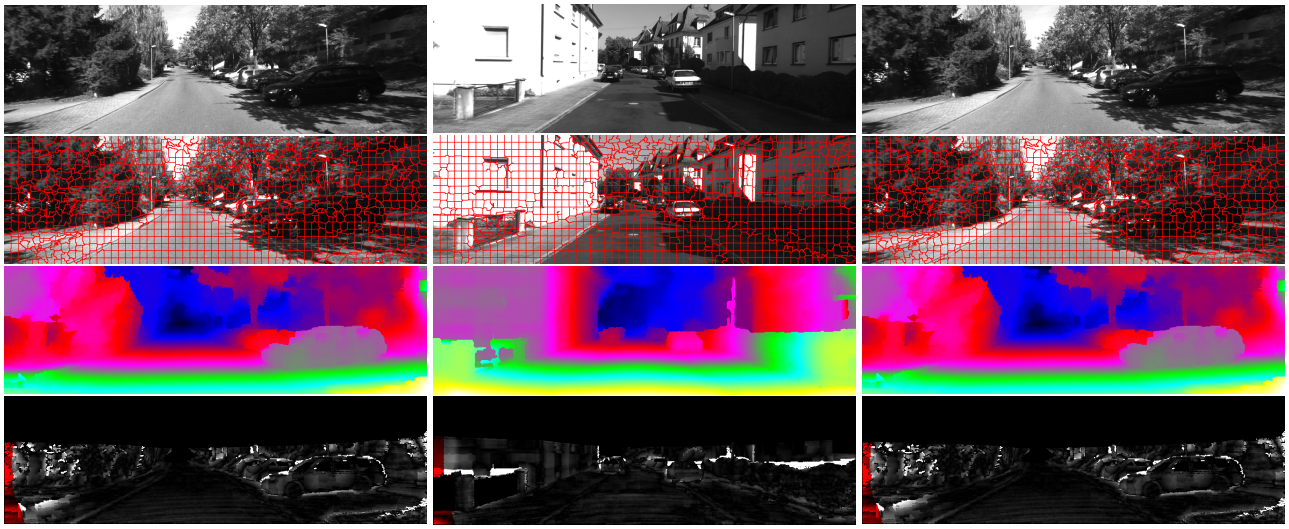


Figure 11. **KITTI**: Our results on the KITTI dataset. Top row shows the image, second row the super pixels, third row the disparity estimates while the errors are displayed in the last row.

Qualitative Results: Qualitative results are shown in Fig. 11. Our approach estimates correct disparity even for difficult cases with shadows and uniform intensity.

6. Conclusion

In this paper, we tackled the problem of unsupervised segmentation with superpixels with the emphasis on speed and accuracy. We build on the work of [31], and propose an efficient coarse to fine optimization technique that is able to converge to a better minimum of the labeling energy in a single iteration. This makes our approach significantly faster. We outperform past work on the BSD dataset, and achieve the lowest NOC error for stereo estimation on the popular KITTI benchmark.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012. 1, 2, 3, 7

[2] A. Ayvaci and S. Soatto. Motion segmentation with occlusions on the superpixel graph. In *Wrk. on Dynamical Vision*, 2009. 1

[3] C. Banz, H. Blume, and P. Pirsch. Real-time semi-global matching disparity estimation on the gpu. In *ICCV Workshops*, pages 514–521. IEEE, 2011. 4

[4] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *CVPR*, volume 1, pages 489–495. IEEE, 1999. 4

[5] J. Chang, D. Wei, and J. W. F. III. A video representation using temporal superpixels. In *CVPR*, 2013. 1

[6] K. V. de Sande, J. Uijlings, T. Gevers, and A. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, pages 1879–1886. IEEE, 2011. 1

[7] M. V. den Bergh, X. Boix, G. Roig, B. de Capitani, and L. V. Gool. Seeds: Superpixels extracted via energy-driven sampling. In *ECCV*, volume 7, pages 13–26, 2012. 1, 2, 7

[8] M. V. den Bergh, G. Roig, X. Boix, S. Manen, and L. V. Gool. Online video superpixels for temporal window objectness. In *ICCV*, 2013. 1, 2

[9] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 2

[10] P. F. Felzenszwalb. Efficient graph-based image segmentation. In *IJCV*, 2004. 6

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 6

[12] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 2

[13] S. Hermann and R. Klette. Iterative semi-global matching for robust driver assistance systems. In *ACCV*, 2012. 7

[14] S. Hickson, S. Birchfield, I. Essa, and H. Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In *CVPR*, 2014. 2

[15] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*, volume 2, pages 807–814. IEEE, 2005. 4

[16] G. Kuschk and D. Cremers. Fast and accurate large-scale stereo reconstruction using variational methods. In *ICCV Workshop on Big Data in 3D Computer Vision*, 2013. 7

[17] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. In *PAMI*, 2009. 2

[18] M. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *CVPR*, 2011. 1

[19] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001. 1, 6

[20] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *CVPR*, 2008. 2

[21] P.Arbelaez, M.Maire, C.Fowlkes, and J.Malik. Contour detection and hierarchical image segmentation. In *PAMI*, 2011. 1, 6

[22] R. Ranftl, T. Pock, and H. Bischof. Minimizing TGV-based Variational Models with Non-Convex Data terms. In *ICSSVM*, 2013. 7

[23] R. Spangenberg, T. Langner, and R. Rojas. Weighted semi-global matching and center-symmetric census transform for robust driver assistance. In *CAIP*, 2013. 7

[24] R. Spangenberg, T. Langner, and S. A. R. Rojas. Large scale semi-global matching on the cpu. In *Intelligent Vehicles Symposium Proc.*, pages 195–201, 2014. 4

[25] J. Tighe and S. Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. *IJCV*, 101(2):329–349, 2013. 1

[26] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *ECCV*, 2008. 2

[27] C. Vogel, S. Roth, and K. Schindler. Piecewise rigid scene flow. In *ICCV*, 2013. 2

[28] K. Yamaguchi, T. Hazan, D. McAllester, and R. Urtasun. Continuous markov random fields for robust stereo estimation. In *ECCV*, 2012. 1, 2, 4, 7

[29] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *CVPR*, 2013. 2

[30] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *CVPR*, 2013. 7

[31] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, 2014. 1, 2, 3, 4, 5, 6, 7, 8

[32] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 1

[33] Y. Zhang, R. Hartley, J. Mashford, and S. Burn. Superpixels via pseudo-boolean optimization. In *ICCV*, 2011. 1