# Sparse Non-local CRF

Olga Veksler
University of Waterloo, Canada
oveksler@uwaterloo.ca

Yuri Boykov
University of Waterloo, Canada
yboykov@uwaterloo.ca

## Abstract

*CRF is a classical computer vision model which is also useful for deep learning. There are two common CRF types: sparse and dense. Sparse CRF connects only the nearby pixels, while dense CRF has global connectivity. Therefore dense CRF is a more general model, but it is much harder to optimize compared to sparse CRF. In fact, only a certain form of dense CRF is optimized in practice, and even then approximately. We propose a new sparse non-local CRF: it has a sparse number of connections, but it has both local and non-local ones. Like sparse CRF, the total number of connections is small, and our model is easy to optimize exactly. Like dense CRF, our model is more general than sparse CRF due to non-local connections. We show that our sparse non-local CRF can model properties similar to that of the popular Gaussian edge dense CRF. Besides efficiency, another advantage is that our edge weights are less restricted compared to Gaussian edge dense CRF. We design models that take advantage of this flexibility. We also discuss connection of our model to other CRF models. Finally, to prove the usefulness of our model, we evaluate it on the classical application of segmentation from a bounding box and for deep learning based salient object segmentation. We improve state of the art for both applications.*

## 1. Introduction

Many computer vision tasks produce so called pixel labelings, where each pixel is assigned a certain label from a pre-defined set. CRF is a classical model [4, 14, 23, 28] that allows enforcing various desired properties on a pixel labeling. CRFs were used for various applications in computer vision prior to deep learning [5, 8, 12, 22, 40, 48, 49] and they are also used in conjunction with deep learning [2, 9, 10, 15, 17, 18, 20, 24, 43, 47, 61].

To model the desired properties for a pixel labeling, one designs a set of interactions between pixels. Most often, these interactions are between pairs of pixels, and they are referred to as edges. These types of CRF are called pairwise. In this paper we address Potts pairwise CRF [38],

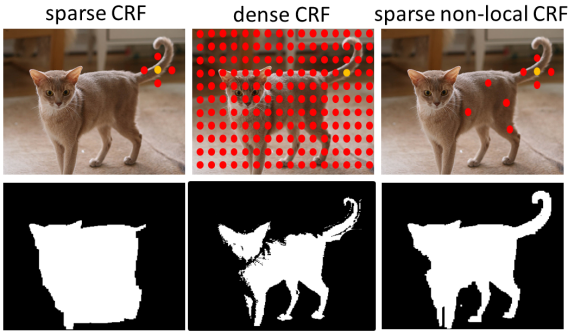



Figure 1. Top row: edge connectivity for sparse, dense and sparse non-local CRF. For clarity, we show neighbors (in red) only for one pixel (in yellow). Bottom row: bounding box segmentation results. Sparse CRF cuts off the thin cat tail. Dense CRF strongly connects the tail with the rest of the cat, and therefore preserves it. Our sparse non-local CRF better connects the tail to the rest of the cat using only a sparse set of connections, and also preserves it.

perhaps the most frequently used CRF model. Potts CRF encourages pixels connected by an edge to be assigned to the same label. We focus on binary pixel labelings, i.e. the set of labels has size two. We call these labels the object and the background, as binary CRFs are often used for object segmentation. Binary pairwise Potts CRF can be optimized exactly and efficiently with a graph cut [8].

In designing Potts CRF, we want to connect pixels that are likely to have the same label. There are two common connectivity types: sparse and dense. As nearby pixels are likely to be in the same object, sparse CRF connects a pixel to its immediate neighbors, usually on a 4 or 8 connected grid, Fig. 1, left. The weight of the edge usually depends on the color similarity between the pixels, since object boundaries tend to cause image edges. The properties of sparse CRF are well understood, it encourages objects that align with image edges and have a shorter boundary [7]. Sparse CRF is known to have a shrinking bias [49], it tends to cut off thin object parts and fill in narrow gaps, Fig. 1, bottom left, preferring a shorter boundary.

To preserve thin parts, we need to connect them to the rest of the object stronger. Dense CRF is based on ob-

servation that not only nearby pairs, but any pair of pixels similar in color and/or position are more likely to have the same label. Therefore, dense CRF connects each pixel to all other pixels, Fig. 1, middle, and the edge strength depends on similarity in color and proximity. This highly connected CRF is expensive to optimize, and only Gaussian edge dense CRF [22] is optimized, approximately, in practice. We will refer to Gaussian edge dense CRF as dense CRF. Dense CRF preserves details better, Fig. 1, middle.

We observe that to preserve details, it is not necessary to connect an object pixel to every other object pixel. A sparser set of non-local connections between the object pixels is sufficient. Of course, we do not know in advance which pixels belong to the object. As pixels of similar color are more likely to be in the same object, these additional non-local connections are chosen (sampled randomly) so that they connect pixels of similar color. Intuitively, given a pixel, to construct the connections, instead of casting a wide "net" to all other image pixels like dense CRF, we cast a targeted "net" towards pixels of similar color. Another argument for a targeted net is that dense CRF uses limited width kernels and many weights are negligible anyway.

We develop a new *sparse non-local* CRF. It has local connections like sparse CRF, and also sparse non-local connections, randomly sampled from pixels of similar color. Our model has detail preserving properties like dense CRF, Fig. 1, right, but achieves this with a sparse set of connections. Therefore our model is optimized globally and efficiently [8]. Furthermore, our model is easier to interpret than dense CRF. We discuss connections between our model and dense CRF [22], OneCut [44], $P^n$-Potts [19].

We validate our model experimentally on traditional segmentation from a bounding box and on deep-learning salient object segmentation weakly supervised with image tags, improving the state of the art in both applications.

## 2. Preliminaries: Binary Pairwise Potts CRF

The main task in CRF is to assign a label $x_p$ to each image pixel $p$. We assume $x_p \in \{0, 1\}$, where 0 is the background and 1 is the object. Let $\mathcal{P}$ be the set of all image pixels, and let $\boldsymbol{x} = (x_p \,|\, p \in \mathcal{P})$ be the labels assigned to pixels in $\mathcal{P}$. For each pixel $p$ there is a unary potential $u_p(x_p)$ which is small if label $x_p$ is likely for $p$ and large otherwise. For binary segmentation, one way to obtain unary terms is to ask a user for object and background seeds [5]. Any pixel $p$ covered by an object seed is hard-constrained to be the object by setting $u_p(0) = \infty$. Any pixel covered by a background seed is similarly constrained to the background. Perhaps the most popular method, widely used and implemented is GrabCut [40]. The user provides a bounding box for the object. Pixels outside the box are hard-constrained to the background, and object/background appearance is modeled from the inside and outside the box for the unary terms.

In addition to the unary terms, there is a pairwise potential for each pair of interacting pixels $p, q$

$$v_{pq}(x_p, x_q) = w_{pq} \cdot [x_p \neq x_q], \tag{1}$$

where $w_{pq} > 0$ and $[x_p \neq x_q]$ is equal to 1 whenever pixels $p, q$ are not assigned to the same label, and 0 otherwise. Pairwise potentials encourage interacting pixels to have the same label. Usually $w_{pq}$ is larger for pixels of similar color.

The best labeling $\boldsymbol{x}$ is found by minimizing the energy

$$E(\boldsymbol{x}) = \sum_{p \in \mathcal{P}} u_p(x_p) + \sum_{(p,q) \in \mathcal{N}} v_{pq}(x_p, x_q), \tag{2}$$

where $\mathcal{N}$ has all interacting pixels pairs. Usually $\mathcal{N}$ is a set. For the simplicity of derivation in Sec. 3, we allow $\mathcal{N}$ to be a multiset, i.e. it can have repeated elements.

When $w_{pq} \geq 0$ the energy in Eq. (2) can be optimized exactly by a graph cut [8]. One sets up a graph with nodes corresponding to image pixels, and edges corresponding to pixel pairs in $\mathcal{N}$. The edge weights are derived from the pairwise terms in Eq. (1). In addition, there are two special nodes called terminals $s, t$. All pixel nodes are connected to $s, t$ by an edge with weight derived from the unary potential $u_p$. The optimal solution is found by a min-cut/max-flow algorithm [6], see [8] for more detail. If $\mathcal{N}$ is sparse, then optimization is efficient. If $\mathcal{N}$ is dense, then optimization, while still polynomial, is impractical, as the size of $\mathcal{N}$ is quadratic in the number of image pixels. Dense CRF [22] uses mean field [21] to optimize Eq. (2) approximately.

## 3. Non Local CRF

In our sparse non-local CRF, we have both the standard 4-connected grid edges, and a sparse set of non-local edges. We rewrite the energy in Eq. (2) to reflect this

$$E(\boldsymbol{x}) = \sum_{p \in \mathcal{P}} u_p(x_p) + \sum_{(p,q) \in \mathcal{N}_l} v_{pq}(x_p, x_q) + \sum_{(p,q) \in \mathcal{N}_{nl}} v_{pq}(x_p, x_q), \tag{3}$$

where $N_l$ is a set of local edges, and $N_{nl}$ is a multiset of non-local edges. A multiset allows element repetition.

We describe the construction of $N_{nl}$ in Sec. 3.1. In Sec. 3.2 we discuss the properties of our sparse non-local CRF. We discuss the connection of our model to dense CRF [22], OneCut [44] and [19] in Sec. 3.3.

### 3.1. Choosing Non-local Edges

Our main idea is that to preserve fine details, it is not necessary to connect each pixel densely. Instead, given a pixel, we can find and connect it to a sparse subset of pixels that are likely to have the same label. Given an object pixel, it is likely that there are many other pixels of similar color

| bins per channel | same label (%) | dist | density (%) |
|---|---|---|---|
| 16 | 92.7 | 106.6 | 76.2 |
| 32 | 94.3 | 99.3 | 75.7 |
| 64 | 95.2 | 93.4 | 71.0 |
| 128 | 96.4 | 88.9 | 57.3 |

Table 1. Experimental evaluation of how likely non-local edges are to stay within the object. See text for description.

in the same object. However there can be many pixels of similar color, and connecting to all of them will not result in a sparse CRF, leading to a costly run time.

Instead, our approach is to sample a sparse subset from the set of pixels with similar color. To sample efficiently, we quantize the color image into bins of equal width in each channel. If two pixels are in the same bin, their color is similar. For each pixel, we uniformly sample a desired number of pixels from its color bin for the non-local edges.

In Tab. 1 we validate experimentally the connection between labels and colors on GrabCut [40] dataset. For each $p$, we draw one random pixel $q$ from the same color bin as $p$. We remove any self-edges and edges between nearest neighbors. In column 3, We compute the average percentage of cases when pixels $p, q$ have the same label. We repeat for different number of bins. The percentage of pixels that have the same label increases with the number of bins, but even with 16 bins, this percentage is larger than 90. This is a strong evidence that choosing at random from quantized color bins is effective for connecting the same label pixels.

In column 4 we compute the average edge length. It increases with an increased number of bins, but is large for all cases, indicating that our edges are non-local. Finally, in the last column of Tab. 1 we show edge density, defined as the percentage of edges left after we remove the nearest neighbors and self-edges. As the number of bins increases, the number of pixels inside each bin gets smaller, and we are more likely to sample the originating pixel or its neighbor. The density drops drastically after 64 bins. In practice, we use 32 or 64 bins, it offers a good balance between having a high percentage of same-class pixels and a good density.

Next we question whether the random nature of the edges effects the results negatively, as each time we apply our CRF, the non-local edge multiset $\mathcal{N}_{nl}$ is different. However, since the majority of the edges connect the same label pixels, the difference in the result is small. Our goal is to create enough non-local connections inside an object to preserve detail, but these connections do not have to be regular. For our deep learning application, edge randomness may even be beneficial as it provides a more diverse training set. Note that randomness is frequently used to improve CNN training, for example dropout [42]. We evaluate the effect of edge randomness on applications in Sec. 4. Here we provide an illustration in Fig. 2. It shows non-local edges chosen for a highlighted (green) pixel on four consecutive runs of segmentation from a bounding box. Non-local edges vary significantly, but the results are almost identical.

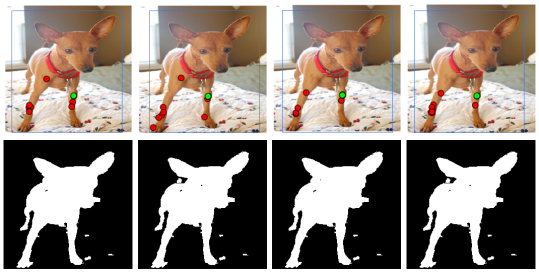


Figure 2. Top row: non-local random edges for the green pixel on four consecutive runs of bounding box segmentation. Bottom row: corresponding segmentation result. Despite a significant variation in connectivity, the results are almost identical.

The next question is how many edges to sample for each pixel. More edges connect stronger, but increase computation. From applications in Sec. 4 we find that from 2 to 8 edges works well, and more edges is not necessarily better.

When sampling non-local edges, we can use more than one image quantization, and for each pixel, sample edges from each quantization. The advantage is that random edges from more than one quantization are more interconnected inside an object that spans a large set of smoothly varying colors. Indeed, with a single quantization, random edges form a disconnected graph. Of course, with local edges, all pixels are inter-connected. Still, there are examples a disconnectedness random edges causes problems that can be fixed with two quantizations, see supplementary materials.

Finally, we note that sampling edges is used in other work, for example for randomized minimum graph cut computation [16], or for graph sparsification [34].

### 3.2. Sparse non-local CRF properties

Since our non-local edges are randomized, given a labeling $\boldsymbol{x}$, we should talk about the expected value of the energy $E(\boldsymbol{x})$ in Eq. (2). The unary terms and the local pairwise terms are not randomized. Furthermore, the properties of the local pairwise terms are well understood [7], they encourage segments with shorter boundary length. Below we discuss the properties of our randomized non-local connections. Let $E^{nl}(\boldsymbol{x})$ be the part of the energy in Eq. (3) that comes from non-local pairwise terms

$$E^{nl}(\boldsymbol{x}) = \sum_{(p,q)\in\mathcal{N}_{nl}} v_{pq}(x_p, x_q), \tag{4}$$

We will derive its expected value, $\bar{E}_{nl}(\boldsymbol{x})$. Let $I_{pq} \in \{0, 1\}$ be an indicator random variable. When we sample edges at pixel $p$, if pixel $q$ is selected then $I_{pq} = 1$. Otherwise $I_{pq} = 0$. Let us denote the expected value of $I_{pq}$ by $\bar{I}_{pq}$.

It is sufficient to consider the case of one particular quantization, as multiple quantizations result in a sum of the corresponding energy expectations. Similarly, it is sufficient to consider the case of one random edge per pixel. Suppose there are $m$ color bins indexed by $1, ..., m$. Let $B_j$ be the set of pixels in bin $j$. We have $\bar{I}_{pq} = Pr(I_{pq} = 1)$. Since pixel $q$ is sampled from the same bin where $p$ is located, if pixels $p, q$ are not in the same bin, then $Pr(I_{pq} = 1) = 0$. Otherwise, $Pr(I_{pq} = 1) = \frac{1}{n_j}$, where $n_j$ is the number of pixels in $B_j$. Given a labeling $\boldsymbol{x}$, the expected non-local pairwise energy is

$$\begin{aligned}\bar{E}^{nl}(\boldsymbol{x}) &= \sum_{p\in\mathcal{P}}\sum_{q\in\mathcal{P}} \bar{I}_{pq} v_{pq}(x_p, x_q) \\ &= \sum_{j\in\{1,...,m\}} \sum_{p\in B_j} \sum_{q\in B_j} \frac{v_{pq}(x_p, x_q)}{n_j} \\ &= \sum_{j\in\{1,...,m\}} \sum_{p\in B_j} \sum_{\substack{q\in B_j \\ x_q\neq x_p}} \frac{w_{pq}}{n_j}, \end{aligned} \tag{5}$$

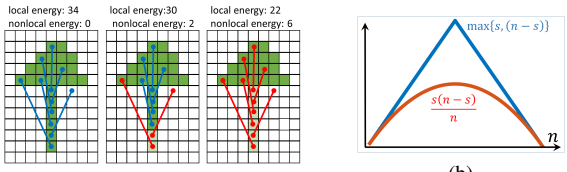
where the last equality holds since we assumed Potts model (Sec. 2), so $v_{pq}(x_p, x_q) = 0$ if $x_p = x_q$, and $w_{pq}$ otherwise.

Eq. (5) has the following intuitive interpretation. Since $w_{pq}$ large for similar pixels, $w_{pq}$ is a measure of similarity between $p, q$. The innermost sum is for a fixed $p$ in bin $j$, and it adds up the similarity of $p$ and all other pixels in bin $j$ that do not have the same label. This sum is normalized by the number of pixels in bin $j$. Thus the inner sum is low when pixels in bin $j$ with a label different from $p$ are the least similar to $p$. The outermost two sums add up these quantities (organized by bins) over all pixels in the image. Thus the expected energy is lower when the two parts (object and background) are dissimilar in colors.

Fig. 3 illustrates how our model preserves details. It shows a green object with the main part and a thin "tail". Three labelings are shown: the first one preserves the object, the second and last remove two and six tail pixels, respectively. With just local edges, pairwise energy decreases from left to right, and so if the unary terms are weak, the last labeling that erases the tail is preferred. For non-local edges, the pairwise energy increases from left to right, counteracting the shrinking bias of local terms. Depending on their relative weight, the tail may be preserved. Note that we put all non-local connections from the tail to the main object part. Since the main part is usually larger and edges are sampled uniformly, it is likely that the tail connects mostly to the main part. Herein is a key to preserving thin parts with a sparse set of non-local connections: they are sparse but likely to connect to where it is important to connect.

### 3.3. Connection to Other CRF Models

**Connection to Dense CRF**




(a) (b)

Figure 3. (a) illustrates how non-local connections help to preserve thin structures, $w_{pq} = 1$ for simplicity. (b) color separation terms

We now discuss connection to dense CRF [22]. Despite its wide use, the properties of dense CRF are not well understood. There is some analysis in [12, 46], but it is under certain assumptions. By relating our model to dense-CRF, we can understand the properties of dense CRF better.

Dense CRF connects every pair of pixels $p, q$ with a weight which is Gaussian in the color and position of $p, q$. Let us quantize an image as in our approach, and let us consider only the edges of dense CRF that are between pixels in some fixed bin. Let the set of pixels in that bin be $B$, and its size be $n$. Suppose we have some labeling $\boldsymbol{x}$. Let $E^d_B(\boldsymbol{x})$ be the pairwise dense CRF energy of this labeling restricted only to the pixels in $B$ and let $\bar{E}^{nl}_B(\boldsymbol{x})$ be our expected non-local energy in Eq. (5) when restricted only to the pixels in $B$, derived similar to Eq. (7):

$$E^d_B(\boldsymbol{x}) = \sum_{p\in B_j} \sum_{\substack{q\in B \\ x_q\neq x_p}} w_{pq} \quad \text{vs.} \quad \bar{E}^{nl}_B(\boldsymbol{x}) = \sum_{p\in B_j} \sum_{\substack{q\in B \\ x_q\neq x_p}} \frac{w_{pq}}{n}.$$

These expressions are identical, except we normalize by $n$. Both discourage splitting bin $B$ among different labels, but if a split happens, dissimilar parts are preferred. Our criterion does not grow as large due to normalization by $n$, similar to normalized cut [41]. So when it is indeed necessary to split a bin, our criterion allows it more easily, compared to dense CRF. The difference is most pronounced if all the edge weights are approximately equal. In this case, dense CRF has a quadratic penalty in $n$ for splitting bin $B$ into two equal parts, while our penalty is linear, less costly.

The downside of normalization by $n$ is that separating one or a few pixels from the rest of the bin is cheap if $n$ is large, even if these pixels are similar to the rest of the bin. Thus our results could potentially be more noisy, but this noise can be counteracted by the local pairwise terms in $\mathcal{N}_l$.

**Connection to OneCut**

OneCut [44] proposes an energy similar to that in GrabCut [40] but tractable. GrabCut energy consists of two parts. The first part is a standard sparse CRF that encourages image edge alignment. The second part encourages an object and background to be equal in size and have a small appearance overlap. This second part is NP-hard, and is optimized iteratively and approximately by GrabCut. OneCut replaces the second hard to optimize part with a color sep-

aration term that encourages an object and background to have a small appearance overlap, and a ballooning term to prevent a trivial (empty object) solution. These new terms, together with the standard sparse CRF term, can be optimized exactly without iteration, in "one cut".

The color separation term in OneCut is based on quantizing an image. Let an image be quantized into $m$ bins indexed from 1 to $m$. Let the size of bin $j$ be $n_j$, and let $s_j(\boldsymbol{x})$ be the number of pixels in bin $j$ that have label 1 in labeling $\boldsymbol{x}$. OneCut color separation term is

$$E^{cs}(\boldsymbol{x}) = \sum_{j\in\{1,\dots,m\}} \min\{s_j(\boldsymbol{x}), n_j - s_j(\boldsymbol{x})\}. \quad (6)$$

For each bin, the smallest cost is when all pixels are assigned to the same label, and the largest cost is when the bin is equally split between the object and background.

Now we turn to our expected non-local energy in Eq. (5). Let us set $w_{pq} = 1$ for all non-local edges. Then Eq. (5) is

$$\begin{aligned}\bar{E}^{nl}(\boldsymbol{x}) &= \sum_{j\in\{1,\dots,m\}} \Big( \sum_{\substack{p\in B_j \\ x_p=0}} \sum_{\substack{q\in B_j \\ x_q=1}} \frac{1}{n_j} + \sum_{\substack{p\in B_j \\ x_p=1}} \sum_{\substack{q\in B_j \\ x_q=0}} \frac{1}{n_j} \Big) \\ &= \sum_{j\in\{1,\dots,m\}} \Big( \sum_{\substack{p\in B_j \\ x_p=0}} \frac{s_j(\boldsymbol{x})}{n_j} + \sum_{\substack{p\in B_j \\ x_p=1}} \frac{n_j - s_j(\boldsymbol{x})}{n_j} \Big) \\ &= 2 \cdot \sum_{j\in\{1,\dots,m\}} \frac{s_j(\boldsymbol{x})(n_j - s_j(\boldsymbol{x}))}{n_j} \quad (7)\end{aligned}$$

In Fig. 3(b) we plot $\frac{s_j(\boldsymbol{x})(n_j - s_j(\boldsymbol{x}))}{n_j}$ in red, dropping subindexes and $\boldsymbol{x}$ for clarity, together with Eq. (6) in blue. Clearly, Eq. (7) is also a color separation term, of a slightly different shape, and similarly to Eq. (6) it encourages object and background to have small appearance overlap.

Thus with $w_{pq} = 1$ for non-local connections, our expected energy is almost the same as in [44]. However, our $w_{pq}$ can vary, and we can design color separation terms that are more general than the cardinality-only ones in Eqs. (6) and (7). Note that the construction in [44] does not allow more general color separation terms. Thus we generalize and improve OneCut, see Sec. 4.2.

**Connection to $P^n$ Potts** In [19] they develop $P^n$ Potts model, which is a high-order CRF. Although our model is pairwise, it is related to $P^n$ Potts. In [19] the motivation is to find spatially coherent clusters and penalize cuts across clusters. In contrast, our motivation is to connect each pixel to a sparse set of color neighbors that are not necessarily spatially close. However, we can interpret our model as penalizing cuts across our (non-local) clusters as well, Eq. (5). It is important to note that our penalty for cutting a cluster is different from [19]. First of all, it is probabilistic and has only expected cost. Moreover, that cost depends on dissimilarity between the two parts: the penalty is lower when two parts are dissimilar (according to pairwise differences). In contrast, the cost in [19] is deterministic and it depends on the cardinality of two parts. The $P^n$ Potts in [19] cannot model dissimilarity of two parts.

## 4. Experimental Results

We now evaluate our sparse non-local CRF on three applications. Two applications are in the classical setting of segmenting an object from a bounding box, GrabCut, Sec. 4.1 and OneCut, Sec. 4.2. The last is a deep learning application for weakly supervised (no pixel precise ground truth) salient object segmentation, Sec. 4.3.

### 4.1. GrabCut

GrabCut [40] segments an object from its bounding box. It starts by modelling the unary terms in Eq. (2) for the object from the inside and the background from the outside of the box using negative log-likelihood of GMM. Then the energy in Eq. (2) is minimized, and the unary terms are re-estimated from the segmented object/background. This process is iterated until convergence.

We use negative log likelihoods of normalized color histogram of quantized image (16 bins) for unary terms [5,48]. For the local edges in $\mathcal{N}_l$ in Eq. (3), we use [5]

$$w_{pq} = \lambda_l \cdot e^{-\frac{\|C_p - C_q\|^2}{2\sigma_{col}^2}}. \quad (8)$$

For non-local edges our first choice is Gaussian weights

$$w_{pq} = \lambda_{nl} \cdot e^{-\frac{\|p-q\|^2}{2\sigma_{pos}^2} - \frac{\|C_p - C_q\|^2}{2\sigma_{col}^2}}, \quad (9)$$

where $C_p$ is the color of pixel $p$. Dense CRF have to use Gaussian weights since their approximate optimization depends on bilateral filtering [36]. We do not have any restriction on edge weights except non-negativity. Our second choice, which we call *distance* weights is

$$w_{pq} = \lambda_{nl} \cdot \frac{1}{\|p-q\|^2} e^{-\frac{\|C_p - C_q\|^2}{2\sigma_{col}^2}}. \quad (10)$$

With distance weights $w_{pq}$ decreases less quickly as the edge length increases, compared to Gaussian weights in Eq. (9). To remove sensitivity to image size, we normalize all coordinates to the range $(1, 100)$.

When optimizing the energies of the type in Eq. (2), it is important to set the relative weights of the unary and pairwise terms appropriately. Choosing the right balance is not trivial and may be image dependent. If the unary terms are unreliable, more weight should be placed on the pairwise terms, but too much weight may result in an empty solution with everything assigned to the background.

We develop a method for avoiding an empty solution. Let $\boldsymbol{x}^0$ be an empty labeling, i.e. $x_p^0 = 0$ for all $p$. Let $\hat{\boldsymbol{x}}$

| # edges | GrabCut Gaussian $w_{pq}$ one quant | GrabCut Gaussian $w_{pq}$ two quant | GrabCut Distance $w_{pq}$ one quant | GrabCut Distance $w_{pq}$ two quant | OneCut |
|---|---|---|---|---|---|
| 2 | .920 (.0023) | .921 (.0031) | .922 (.0035) | .926 (.0024) | .905 (.0014) |
| 4 | .922 (.0019) | .920 (.0032) | .923 (.0028) | .927 (.0011) | .905 (.0015) |
| 8 | .920 (.0022) | .920 (.0032) | .926 (.0029) | .928 (.0012) | .906 (.0014) |
| 16 | .919 (.0029) | .919 (.0028) | .925 (.0030) | .928 (.0007) | .908 (.0014) |
| 32 | .919 (.0026) | .920 (.0029) | .925 (.0027) | .928 (.0005) | .908 (.0014) |

Table 2. Experimental evaluation of the variation in performance for GrabCut and OneCut. Mean $F_\beta$ scores (higher is better) over 20 trials for the GrabCut dataset, and std in parenthesis.

be a reasonable non-empty solution. We want to ensure that the parameter setting is s.t. our energy in Eq. (3) is lower for a reasonable non-empty labeling than for an empty one, i.e. $E(\hat{\boldsymbol{x}}) < E(\boldsymbol{x}^0)$. If this holds, we proceed to optimization. If $E(\hat{\boldsymbol{x}}) > E(\boldsymbol{x}^0)$, we add a "balooning" term to our energy

$$E^b(\boldsymbol{x}) = \lambda_b \sum_{p \in \mathcal{P}} (1 - x_p).$$

where $\lambda_b$ is just sufficiently large to ensure $E(\hat{\boldsymbol{x}}) + E^b(\hat{\boldsymbol{x}}) < E(\boldsymbol{x}_0) + E^b(\boldsymbol{x}_0)$. Computing such $\lambda_b$ is a simple algebraic manipulation, given in the supplementary materials.

To find a reasonable non-empty $\hat{\boldsymbol{x}}$, we sort pixels inside the box by their preference for the foreground, and hard-constraint the top $r$ fraction of them to the foreground. In practice, we set $r = 1/6$. Then we perform optimization of Eq. (3) and the resulting solution is $\hat{\boldsymbol{x}}$.

Our empty solution avoidance is implementable whenever it is possible to compute an exact energy of a labeling and to find a globally optimal solution. Therefore we also apply empty solution avoidance to our re-implementation of GrabCut with sparse CRF, to ensure that our improvement is not just due to avoiding empty solutions. For dense CRF [22], one cannot efficiently compute the exact labeling energy, so our empty solution approach is not applicable.

We now discuss parameter settings. Prior work [12, 40, 44] chooses parameters that work well for the grabcut dataset [40], and with the exception of [12], they report performance only on the GrabCut dataset. Since there are only a handful of parameters, the overfitting problems with such approach are minor, at least compared to deep learning. But an overfitting does happen. For a fair comparison to prior work, we also tune parameters on GrabCut dataset, but we report performance on two other datasets, keeping parameters fixed to those tuned on GrabCut dataset.

For evaluation we use GrabCut dataset [40], which has 50 images with bounding boxes. We also use MSRA1K [1] and ECSSD [53] datasets, 1,000 images each. These are salient object datasets without bounding boxes. As in [12], we construct boxes from the ground truth segmentations. ECSSD is more challenging than the other two datasets.

Since our approach is based on random edges, we test the variation in performance over different runs, and the dependence on the number of edges sampled per pixel, and on one or two quantizations. When we increase the number of edges, we divide $\lambda_{nl}$ in Eqs. (9) and (10) by the number of edges to keep the overall relative weight of non-local pairwise terms the same. We use GrabCut dataset, and test both Gauss and distance edges, Eqs. (9) and (10). In each case, we perform 20 runs over the dataset and compute the mean and std of $F_\beta$. The results are in Tab. 2. The std is low in all cases, which means that our results are stable over different runs. For Gaussian weights, there is no significant variation in performance between one or two quantizations and the number of edges. For distance edges, there is an improvement when using more edges, up to 16. The results with two quantizations are also better than with one quantization. For further experiments, we use 2 quantizations, 2 edges for Gaussian, and 8 edges for distance weights.

In Tab. 3, left, we compare to methods that use $F_\beta$ metric[1] . GrabCut$_1$ is the original method [40], with GMM modeling. GrabCut$_2$ is our own implementation of GrabCut, with the same unary and local pairwise potentials as in our method, and with our empty solution avoidance strategy. DenseCut$_1$ are the results from [12], who develop a GrabCut algorithm with sparse CRF replaced by dense CRF[2]. DenseCut$_2$ is our implementation of GrabCut with dense CRF instead of sparse CRF. Next we have our sparse non-local CRF GrabCut implementations for Gaussian and distance edges. $P^n$-Potts is the model in [19][3]. Our models are better in all but one case, and significantly better on a more difficult ECSSD dataset.

Dense GrabCut$_2$ and ours (gauss, dist) is a direct comparison between dense CRF and our sparse non-local CRF, as the only difference between these is the CRF type. Our performance may be better due to a better model (less heavy penalty for splitting color bins), or due to better optimization (global minimum vs. approximation), or both.

In Tab. 3, right, we compare to methods that use the error rate in the box. From the traditional methods, only [25] performs better, but they use different boxes more suited for their tight box prior. A deep learning method [52] trained on MSCOCO [30] performs the best, as expected. Our approach is less than 2 points behind [30], despite not requiring pixel precise ground truth.

Qualitative comparisons are in Fig. 4. Observe fine details in our results (d,e), especially compared to sparse CRF (b). Our running time is 1.2 and 2.1 seconds per image with Gaussian and distance edges in matlab implementation. Our

[1] $F_\beta = \frac{(1+\beta^2) precision \times recall}{\beta^2 \times precision + recall}$, with $\beta^2 = 0.3$, as in prior work.

[2] We report their published metrics for GrabCut and MSRA1K datasets, and the result from running their code https://githubmemory.com/repo/Juseong-Bang/densecut--windows-master on ECSSD dataset

[3] In [19], they multiply cardinality by cluster quality learnt from pixel precise ground truth. Using ground truth would be unfair for comparison, so we base cluster quality on color variance. All other aspects are the same.

| | GrabCut | MSRA1K | ECSSD |
|---|---|---|---|
| sparse GrabCut$_1$ | .909 | .945 | NA |
| sparse GrabCut$_2$ | .897 | .956 | .868 |
| dense GrabCut$_1$ | **.932** | .959 | .829 |
| dense GrabCut$_2$ | .872 | .950 | .837 |
| $P^n$ Potts | .911 | .957 | .857 |
| ours (gauss) | .919 | **.966** | **.892** |
| ours (dist) | .928 | .961 | .880 |

| | GrabCut |
|---|---|
| OneCut [44] | 6.7 |
| TightBox [25] | 3.7 |
| Kernel [45] | 7.1 |
| deepCut [52] | **3.3** |
| ours (gauss) | 5.5 |
| ours (dist) | 5.1 |

Table 3. Experimental evaluation of GrabCut algorithm. Left: methods that use performance metric $F_\beta$ score, higher is better, right: methods using error rate in the box.

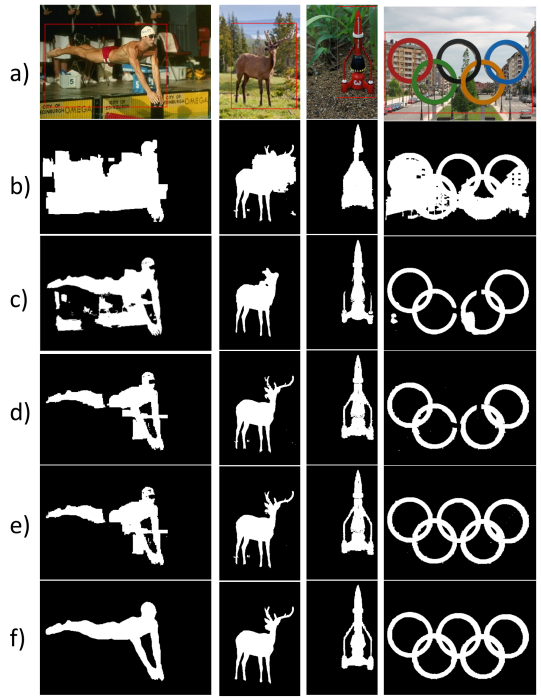



Figure 4. Comparative result for the GrabCut application: a) input image with the bounding box; b) GrabCut with sparse CRF, our implementation; c) DenseCut [12], their implementation; d) our sparse non-local CRF with distance edge weights; e) our sparse non-local CRF with Gaussian edge weights; f) ground truth.

model parameters are in the supplementary materials.

### 4.2. OneCut

As discussed in Sec. 3.3, we can approximate OneCut [44] with our sparse non-local CRF by setting all $w_{pq}$ weights for non-local edges to 1. However, it is more interesting to generalize their color separation term by setting $w_{pq}$ weights to depend on color difference between pixels. This way the color separation term is not just based on cardinality, i.e. the number of pixels split between the labels in a color bin, but also on the color similarity of pixels split across a bin. Intuitively, this makes sense since if we do have to break some color bin, it is better to break it so

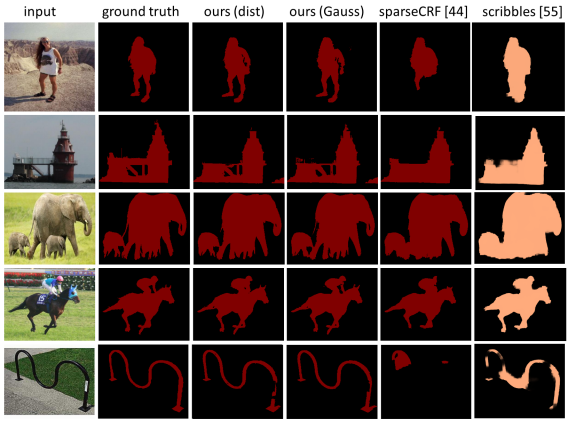



Figure 5. Comparative results for the salient object segmentation. Last column is with scribbles, a stronger form of weak supervision than the other results shown.

that more similar colors stay together. We use the weights in Eq. (8) for the non-local edges, but with a a smaller $\sigma_{col}$, compared to the local edges. The variation of performance depending on the number of edges is in Tab. 2, right. Unlike GrabCut, using more edges is better. We use 16 edges and 64 bins in practice. For the GrabCut dataset, the unmodified OneCut $F_\beta = 0.897$, and for our version with the more general color term, $F_\beta = 0.908$. Running time is 2.3 sec in matlab. For more results, see the supplementary materials.

### 4.3. Salient Object Segmentation

We apply sparse non-local CRF for image tag weakly supervised salient object segmentation based on deep learning. In this setting, a dataset known to contain salient objects is given but without pixel precise ground truth.

We modify the approach in [47]. They develop a loss function that works well for training salient object segmentation CNN without pixel precise ground truth. Sparse CRF is the most important part of their loss function. They also experiment with dense CRF but find that it does not work, perhaps due to the difficulty of minimizing the loss function of their type[4]. We take their approach but replace sparse CRF with our sparse non-local CRF. We use CNN architecture from [47][5], Unet [39] with ResNeXt [51] fixed features pretrained on Imagenet [13] and train $256 \times 256$ images. See the supplementary materials for the details of our modification the loss function in [47] and training parameters.

We use datasets DUTS [50], DUTO [54], ECSSD [53], MSRAB [31], THUR [11], SED2 [3], SOD [33], PascalS [29], and HKU-IS [27]. In addition to $F_\beta$, for fair

[4]Note that using dense CRF in a loss function is different from using dense CRF as part of architecture. Since there is no pixel precise ground truth, having dense CRF in architecture does not help in designing a loss function useful for training in weakly supervised setting.

[5]https://github.com/morduspordus/SingleClassRL

| | MSRAB | | ECSSD | | DUTO | | PascalS | | THUR | | SED2 | | SOD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ |
| SBF [58] | - | - | .787 | .085 | .583 | .135 | .680 | .141 | - | - | - | - | .676 | .140 |
| USD [60] | .877 | .056 | .878 | .070 | .716 | .086 | **.842** | .139 | .732 | .081 | .838 | .088 | .798 | **.118** |
| WSI [26] | .890 | .067 | .837 | .110 | .722 | .101 | .752 | .152 | - | - | - | - | .751 | .185 |
| DeepUSPS [35] | .903 | .040 | .874 | .063 | .736 | **.063** | - | - | - | - | .845 | **.070** | - | - |
| SparseCRF [47] | .885 | .046 | .894 | .056 | .753 | 073 | .833 | .090 | .725 | .078 | .836 | .096 | **.825** | .126 |
| ours (gauss) | .902 | .042 | .898 | **.042** | .767 | .070 | **.842** | **.089** | .729 | .077 | .853 | .090 | .812 | .132 |
| ours (dist) | **.907** | **.039** | **.902** | .056 | **.771** | .070 | .841 | .090 | **.734** | **.075** | **.876** | .080 | .810 | .135 |

Table 4. MSRA training dataset: comparison to other image tag weakly supervised salient object segmentation methods. Performance metrics are $F_\beta$ (higher is better) and $mae$ (lower is better).

| | MSRAB | | ECSSD | | DUTO | | PascalS | | SOD | | DUTS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $maxF_\beta$ | $mae$ | $maxF_\beta$ | $mae$ | $maxF_\beta$ | $mae$ | $maxF_\beta$ | $mae$ | $maxF_\beta$ | $mae$ | $maxF_\beta$ | $mae$ |
| WSS [50] | .877 | .076 | .856 | .104 | .687 | .118 | .778 | .141 | .780 | .170 | - | - |
| MSW [57] | **.890** | .071 | .878 | .096 | .718 | .114 | .790 | .134 | .799 | .167 | - | - |
| sparseCRF [47] | .873 | .055 | .893 | 060 | .756 | .074 | .845 | .087 | .825 | .130 | .800 | .061 |
| ours (gauss) | .875 | **.051** | **.910** | **.049** | .785 | .067 | .861 | **.076** | **.845** | **.115** | .833 | **.050** |
| ours (dist) | .878 | **.051** | .909 | .053 | **.793** | **.066** | **.864** | .079 | .840 | .120 | **.840** | **.050** |

Table 5. DUTS training dataset: comparison to other image tag weakly supervised salient object segmentation methods. Performance metrics are $maxF_\beta$ (higher is better) and $mae$ (lower is better).

| | ECSSD | | DUTO | | PascalS | | HKUIS | | THUR | | DUTS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ | $F_\beta$ | $mae$ |
| scribbles$_1$ [59] | .880 | .061 | .750 | .068 | .813 | .140 | .870 | .047 | .714 | .077 | .777 | .062 |
| scribbles$_2$ [56] | .900 | **.049** | .758 | **.060** | .823 | .078 | .896 | .038 | **.755** | **.069** | .823 | **.049** |
| boxes [32] | .860 | .072 | .686 | .081 | - | - | .853 | .058 | - | - | .736 | .079 |
| ours (gauss) | .905 | **.049** | .781 | .067 | .859 | **.076** | .912 | **.035** | .731 | .075 | .827 | **.049** |
| ours (dist) | **.909** | .053 | .790 | .066 | **.862** | .079 | **.915** | .037 | .745 | .070 | **.839** | **.049** |

Table 6. DUTS training dataset: comparison to [56, 59], who use scribbles, and [32] bounding boxes, both much stronger forms of weak supervision. Performance metrics are $F_\beta$ (higher is better) and $mae$ (lower is better).

comparison with prior work, we also use $maxF_\beta$, the maximum $F_\beta$ across the binary maps of different threshold, and mae [37], the average absolute per pixel difference between the predicted saliency and ground truth. Some prior work trains on MSRAB [31], some on DUTS [50], we train on MSRAB and DUTS, and compare correspondingly.

For fair comparison, we re-train sparseCRF [47] on $256 \times 256$ images, getting results better than those reported in [47], where they train on $128 \times 128$ images. For other prior work, we either use the numbers they report, or run their code. For our sparse non-local CRF, we evaluate both Gaussian and distance weights. The quantitative comparison is in Tab. 4 for methods that train on MSRAB, and in Tab. 5 for methods that train on DUTS. In Tab. 6 we also compare to [56, 59], who use scribbles, and to [32], who use boxes, both much stronger supervision. In most cases, our methods take both the first and second place, even when comparing with stronger supervision. When our models do not take the first place, it is only by a small margin, except for THUR dataset for comparing with scribble supervision

Figure 6. Some failure examples for GrabCut application. From left to right: input image, ground truth, our results, dist weights.

in terms of $F_\beta$. The qualitative results are in Fig. 5. Observe the fine detail preservation. In some cases our results have details more accurate than the ground truth, for example, in the top row, the space between the woman's arm and her body is properly segmented by our methods, but absent in ground truth, and in row 4, we segment the grass peeking through the bend of the horse knee, absent in ground truth.

**Limitations**

Our model is limited to attractive ($w_{pq} > 0$) pairwise potentials, as our sampling strategy would not be effective for repulsive [55] potentials. While effective at preserving fine details, our model, like dense CRF, is more prone than sparse CRF to connect fine spurious details to the object, see Fig. 6. Thin structures similar to object in color are added. Adding spurious fine detail is less of a problem for CNN based salient object segmentation, as object appearance is learned over a large training dataset.

## References


[1] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1597–1604. IEEE, 2009. 6

[2] Jiwoon Ahn and Suha Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 4981–4990, 2018. 1

[3] Sharon Alpert, Meirav Galun, Achi Brandt, and Ronen Basri. Image segmentation by probabilistic bottom-up aggregation and cue integration. *IEEE transactions on pattern analysis and machine intelligence*, 34(2):315–327, 2011. 7

[4] A. Blake, P. Kohli, and C. Rother. *Markov random fields for vision and image processing*. MIT Press, 2011. 1

[5] Yuri Boykov and Gareth Funka-Lea. Graph cuts and segmentation. *IJCV*, pages 109–131, 2006. 1, 2, 5

[6] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *EMMCVPR*, pages 359–374, 2001. 2

[7] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*, pages 26–33, 2003. 1, 3

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001. 1, 2

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International Conference on Learning Research*, 2015. 1

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. 1

[11] Ming-Ming Cheng, NiloyJ. Mitra, Xiaolei Huang, and Shi-Min Hu. Salientshape: group saliency in image collections. *The Visual Computer*, 30(4):443–453, 2014. 7

[12] Ming-Ming Cheng, Victor Adrian Prisacariu, Shuai Zheng, Philip HS Torr, and Carsten Rother. Densecut: Densely connected crfs for realtime grabcut. In *Computer Graphics Forum*, volume 34, pages 193–201. Wiley Online Library, 2015. 1, 4, 6, 7

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 7

[14] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984. 1

[15] Varun Jampani, Martin Kiefel, and Peter V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Conference on Computer Vision and Pattern Recognition*, pages 4452–4461, 2016. 1

[16] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, volume 93, pages 21–30. Citeseer, 1993. 3

[17] Patrick Knöbelreiter, Christian Reinbacher, Alexander Shekhovtsov, and Thomas Pock. End-to-end training of hybrid CNN-CRF models for stereo. In *CVPR*, pages 1456–1465. IEEE Computer Society, 2017. 1

[18] Patrick Knobelreiter, Christian Sormann, Alexander Shekhovtsov, Friedrich Fraundorfer, and Thomas Pock. Belief propagation reloaded: Learning bp-layers for labeling problems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[19] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009. 2, 5, 6

[20] Alexander Kolesnikov and Christoph H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *European Conference on Computer Vision*, pages 695–711, 2016. 1

[21] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models*. The MIT Press, 2009. 2

[22] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Neural Information Processing Systems*, pages 109–117, 2011. 1, 2, 4, 6

[23] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields. In *International Conference on Machine Learning*, 2001. 1

[24] Jungbeom Lee, Eunji Kim, Sungmin Lee, Jangho Lee, and Sungroh Yoon. Ficklenet: Weakly and semi-supervised semantic image segmentation using stochastic inference. In *Conference on Computer Vision and Pattern Recognition*, pages 5267–5276, 2019. 1

[25] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *2009 IEEE 12th international conference on computer vision*, pages 277–284. IEEE, 2009. 6, 7

[26] Guanbin Li, Yuan Xie, and Liang Lin. Weakly supervised salient object detection using image labels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 8

[27] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–487, June 2016. 7

[28] S. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995. 1

[29] Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287, 2014. 7

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[31] Tie Liu, Jian Sun, Nan-Ning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 7, 8

[32] Yuxuan Liu, Pengjie Wang, Ying Cao, Zijian Liang, and Rynson WH Lau. Weakly-supervised salient object detection with saliency bounding boxes. *IEEE Transactions on Image Processing*, 30:4423–4435, 2021. 8

[33] Vida Movahedi and James H Elder. Design and perceptual validation of performance measures for salient object segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 49–56. IEEE, 2010. 7

[34] Jeova F. S. Rocha Neto and Pedro F. Felzenszwalb. Spectral image segmentation with global appearance modeling, 2020. 3

[35] Tam Nguyen, Maximilian Dax, Chaithanya Kumar Mummadi, Nhung Ngo, Thi Hoai Phuong Nguyen, Zhongyu Lou, and Thomas Brox. Deepusps: Deep robust unsupervised saliency prediction via self-supervision. In *Advances in Neural Information Processing Systems*, pages 204–214, 2019. 8

[36] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. *IJCV*, pages 24–52, 2009. 5

[37] Federico Perazzi, Philipp Krähenbühl, Yael Pritch, and Alexander Hornung. Saliency filters: Contrast based filtering for salient region detection. In *Conference on Computer Vision and Pattern Recognition*, pages 733–740. IEEE, 2012. 8

[38] R. Potts. Some generalized order-disorder transformation. *Proceedings of the Cambridge Philosophical Society*, 48:106–109, 1952. 1

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015. 7

[40] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM TOG*, 23:309–314, 2004. 1, 2, 3, 4, 5, 6

[41] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–737, 1997. 4

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, January 2014. 3

[43] Meng Tang, Abdelaziz Djelouah, Federico Perazzi, Yuri Boykov, and Christopher Schroers. Normalized cut loss for weakly-supervised CNN segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 1818–1827, 2018. 1

[44] Meng Tang, Lena Gorelick, Olga Veksler, and Yuri Boykov. Grabcut in one cut. In *ICCV*, 2013. 2, 4, 5, 6, 7

[45] Meng Tang, Dmitrii Marin, Ismail Ben Ayed, and Yuri Boykov. Kernel cuts: Kernel and spectral clustering meet regularization. *International Journal of Computer Vision*, 127(5):477–511, 2019. 7

[46] O. Veksler. Efficient graph cut optimization for full crfs with quantized edges. *Transactions on Pattern Analysis and Machine Intelligence*, 2018. 4

[47] Olga Veksler. Regularized loss for weakly supervised single class semantic segmentation. In *European Conference on Computer Vision*, pages 348–365. Springer, 2020. 1, 7, 8

[48] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Joint optimization of segmentation and appearance models. In *2009 IEEE 12th International Conference on Computer Vision*, pages 755–762. IEEE, 2009. 1, 5

[49] G. Vogiatzis, P.H.S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Conference on Computer Vision and Pattern Recognition*, pages II: 391–398, 2005. 1

[50] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017. 7, 8

[51] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Conference on Computer Vision and Pattern Recognition*, pages 5987–5995, 2017. 7

[52] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. Deep grabcut for object selection. In *BMVC*, 2017. 6, 7

[53] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1155–1162. IEEE, 2013. 6, 7

[54] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *Conference on Computer Vision and Pattern Recognition*, pages 3166–3173. IEEE, 2013. 7

[55] Stella Yu and Jianbo Shi. Segmentation with pairwise attraction and repulsion. In *Proceedings of (ICCV) International Conference on Computer Vision*, pages 52 – 58, Vancouver, British Columbia, July 2001. 8

[56] Siyue Yu, Bingfeng Zhang, Jimin Xiao, and Eng Gee Lim. Structure-consistent weakly supervised salient object detection with local saliency coherence. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 3234–3242. AAAI Press, 2021. 8

[57] Yu Zeng, Yunzhi Zhuge, Huchuan Lu, Lihe Zhang, Mingyang Qian, and Yizhou Yu. Multi-source weak supervision for saliency detection. In *Conference on Computer Vision and Pattern Recognition*, pages 6074–6083, 2019. 8

[58] Dingwen Zhang, Junwei Han, and Yu Zhang. Supervision by fusion: Towards unsupervised learning of deep salient object detector. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4048–4056, 2017. 8

[59] Jing Zhang, Xin Yu, Aixuan Li, Peipei Song, Bowen Liu, and Yuchao Dai. Weakly-supervised salient object detection via scribble annotations. In *Conference on Computer Vision and Pattern Recognition*, pages 12546–12555, 2020. 8

[60] Jing Zhang, Tong Zhang, Yuchao Dai, Mehrtash Harandi, and Richard Hartley. Deep unsupervised saliency detection: A multiple noisy labeling perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9029–9038, 2018. 8

[61] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision*, pages 1529–1537, 2015. 1