Graphical Models Course : Project Report

Chady Raach
chady.raach@student-cs.fr

Marah Gamdou
marah.gamdou@student-cs.fr

Abstract

In this project, we explore two s-t graph cut algorithms that use min-cut for image segmentation. The purpose of the segmentation is to separate the object from the background. Our work is based on two parts : transforming the image into a suitable graph for a max-flow based background-foreground segmentation, testing both algorithms on Oxford-IIIT Pet Dataset and comparing their efficiency on the latter task.

1. Introduction

Image segmentation is the process of partitioning an image into regions or components that are non overlapping. It has interested the computer vision community and several approaches have been proposed from descriptors and low vision features to sophisticated deep learning frameworks.

In this project, we propose a solution for image segmentation problem by modeling an image as a graph and applying an s-t graph cut to find the components of the image. We simplify the problem of image segmentation to finding the background and foreground components. It can be generalized to multi-instance segmentation by using binary segmentation in the context of one-to-many segmentation.

2. Problem Definition

The goal of this project is to do an object/background segmentation based on a s-t graph cut. Once an image is modeled as a graph, we can apply a wide family of algorithms to find the cheapest cut that separates object components and background components. By assigning labels of object and background to each pixel, we minimize the labeling energy of the image and thus solve the segmentation problem. In our case, the cost to minimize is discrete.

In this project, we first explain how to create a graph with a topology that models the image and take into account its structural and local properties. Secondly, we explain the equivalence between the labeling energy minimization and min-cut optimization. Then, we present two different algorithms widely used for min-cut/max-flow solving: Ford-Fulkerson as an augmenting path algorithm and Goldberg-Tanjan as a Push-Relabel algorithm. Finally, we show the results we obtained from some numerical experiments conducted on few images from an image segmentation dataset (with labels).

3. Creating a Graph from an Image

3.1. Description of the graphical structure

Given a square image of characteristic dimension H, we want to build a graph $G = (V, E)$ where V is the set of the graph vertices and E is the set of the graph directed edges.

A pixel is represented by a vertex in the graph. The pixel at the position (i, j) in the image corresponds to the vertex $i \times H + j$. In addition to pixel vertices, we define two additional vertices $\mathbf{s}$ and $\mathbf{t}$ denoting "sink" and "tank" modeling the possible instances "object" and "background".

Connections between vertices are ensured by edges. In our graphical representation, two types of edges are defined: n-links and t-links.

- n-links are connections between pixel vertices. Each pixel vertex is connected to its 4 neighboring pixel vertices by a directed edge. This type of connection transmits the structural and local features of the image since a pixel is strongly impacted by the value and the instance of its neighboring pixel. With this graph topology, the dimension of the solution space is reduced to a small set where the cut is done only between tangent pixels. Hence it can be easily transformed into a solution of the image segmentation problem.

- t-links are connections between pixel vertices and the seed vertices that represent either the object instance or the background instance. This type of connection transfers the a priori knowledge that we have about the membership of certain pixels to each of the instances. These connections allow to easily deduce the instance of the strongly connected components given by the graph cut.

In figure 1, we show an example of a graphical representation of images with t-links and n-links.

To make this graphical representation more accurate, we use the pixel values to depict the similarity and the disconti-

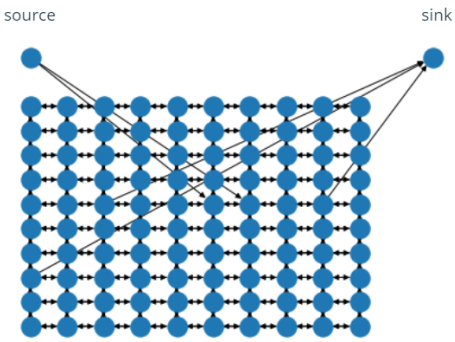


Figure 1: Graphical representation of a 10×10 image

nuity between pixels. The next two subsection explains how this relation is expressed in terms of labeling energy if the problem is formalised as energy minimization, and in terms of edge capacity if the problem is formulated as a min-cut.

3.2. Pairwise/Pixel cost

The pixel cost will represent the flow over the n-links. It thus penalizes discontinuity between different pixels. This cost is considered in our example as the similarity between pixels, as shown in equation 1, where $\sigma = 30$ for gray-scale images (with 255 level of gray).

$$sim(p,q) = exp(-\frac{(I_p - I_q)^2}{2\sigma^2}) \tag{1}$$

3.3. Unary Cost

The unary cost is the one that represents the flow over the t-links. In other words, it defines how much flow the source can supply and the tank can receive. Notice that not all the nodes are connected to the sink and source, therefore in equation 2, we took the maximum over all the n-links flow, to ensure saturation of some n-links in the graph. In our experiments, we will demonstrate that moving the unary connections around the object of interest or the background (while ensuring consistency) does not have a large impact on the segmentation result.

$$UC = \max_{p,q \in E} sim(p,q) \tag{2}$$

4. Maximum Flow Problem

The labeling energy to minimize can be expressed by equation 3, where y_i is the label assigned to the pixel i and c_i is the cost of assigning the label.

$$E = \sum_i c_i y_i + \sum_{i,j} c_{i,j} y_i (1 - y_j) \tag{3}$$

The equivalence between solving the max-flow problem and finding the minimum energy of labelling was shown by Ford Fulkerson. To have some intuition about the result, one can examine limit cases: when two pixels are different, the cost is high while the similarity score is low. Hence, the edge can be easily saturated with flow coming from similar pixel and the two pixels belong to different sets of the cut. Conversely, if the pixels are similar, the edges are unlikely to be saturated because of the high capacity value. The optimal cut minimizes the second term of the energy function which corresponds to the pairwise cost.

4.1. Ford-Fulkerson (1962)

In order to find the maximum flow from the source to the tank, this algorithm searches iteratively an augmenting path and adds the maximum flow that can be transferred through it. At each iteration of the algorithm the following constraints are satisfied:

- capacity constraint: $\forall (u,v) \in E, f(u,v) \leq c(u,v)$
- flow conservation: $\forall u \in V \backslash \{s,t\}, \sum_{v \in E} f(u,v) = 0$

To perform this algorithm, we define a dummy graph called "residual graph" which has the same topology as the graph on which we want to maximize the s-t flow. This graph differs by its edge capacity: once the maximal possible flow is passed through a path, the capacity of an edge is diminished by the added flow if the edge is oriented in the same direction as the graph G. If the edge is visited in the reverse direction, its capacity is augmented by the corresponding flow. The algorithm will iteratively search for a path in the residual graph, and will stop if no possible path from the source to the tank can be found. This means that all the edges are saturated and the flow passed is maximal. We can deduce the graph cut by cutting off the saturated edge. It is possible to end with at least two disjointed strongly connected components since the maximum flow algorithm is terminated which means that all possible paths from s to t have at least one saturated edge.

4.2. Push-Relabel (Goldberg and Tarjan, 1988)

In this min-cut method, instead of maintaining a feasible flow, and improving it until it becomes optimal, we maintain a preflow, which is an assignment of flows to edges that allows vertices to have more incoming flow than outgoing flow (but not vice versa) and that satisfies the capacity constraints.

It is therefore a relaxation algorithm where the flow constrained is partially relaxed temporarily. In fact, local updates are performed repeatedly until the global constraint is satisfied.

This algorithm is mainly based on two operation, as shown in Algorithm 2 :

Algorithm 1 Ford-Fulkerson algorithm

$R_f \leftarrow c$
$f \leftarrow 0$
while there is a path γ between s and t in R_f **do**
  $\Delta \leftarrow min(R_f(u,v); (u,v) \in \gamma)$
  for (u,v) $\in \gamma$ **do**
   if (u,v) $\in E$ **then**
    $f(u,v) = f(u,v) + \Delta$
   else
    $f(u,v) = f(u,v) - \Delta$
   end if
  end for
end while

- Push operation : sending flow to neighbors
- Relabel operation : increasing the height of a node

Algorithm 2 Push-Relabel algorithm

Input network $(G = (V, E), s, t, c)$
Output f
$h[s] = |V|$
for each v in V-{s} do h[v]=0 **do**
  for each $(s, v) \in E$ do f(s,v)=c(s,v) **do**
   while f is not a feasible flow **do**
    let c'(u,v)=c(u,v)+f(u,v)-f(v,u) be the capacities of the residual network
    if there is a vertex $v \in V - \{s, t\}$ and a vertex $w \in V$ such that $e_f(v) > 0$, $h(v) > h(w)$, and $c'(v, w) > 0$ **then**
     push min{c'(v,w), e_f(v)} units to the flow on the edge (v,w)
    else let v be a vertex such that $e_f > 0$ and set h[v]=h[v]+1
    end if
   end while
  end for
end forf

5. Experiments

In this section, we present the experiments we performed to compare the effectiveness of Ford-Fulkerson and Push-Relabel, and to evaluate their performance in the image segmentation task.

5.1. Dataset

For our experiments, we will be using the The Oxford-IIIT Pet Dataset [1] which is an image segmentation dataset with available ground truth annotations and large variations in scale, pose and lighting. However, we will not be using the whole dataset as it is generally used for deep learning based image segmentation.

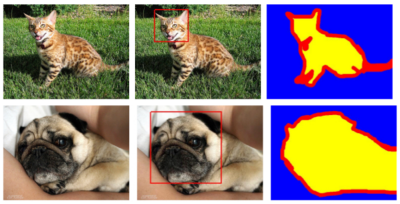


Figure 2: Dataset

5.2. Score

The score that we will be computing for each image is the pixel accuracy, based on the ground truth masks of each image.

5.3. Image segmentation pipeline

Figure 3 shows the pipeline that we used for our image segmentation experiments. The first step in the pipeline is

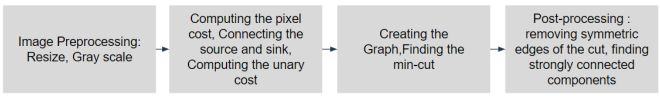


Figure 3: Our pipeline

to transform the image to a grayscale image and resize it to a square, low dimensional image. This reduced image is then used to compute the similarity between the pixels and to deduce the capacity of the n-links and t-links described in the previous sections. The graph is thus constructed and the connections are well defined according to the local and structural properties of the image. We manually choose pixels to be defined as background or foreground by connecting them to the sink or the tank.

To this graph, we apply a max-flow algorithm to define a cut. When the algorithm finishes, we have different strongly connected components: those that are still connected to the seeds are labeled accordingly since they use an a priori knowledge that we passed to the model. The other strongly connected components are labeled by default as background instances. Finally, we post-process the result by up-scaling the mask to the size of the original image.

5.4. Ford-Fulkerson

We have implemented Ford-Fulkerson algorithm and tested it on some images from IIIT Pet dataset. We reported some of our results in Figure 5. Overall, the result of object segmentation is acceptable given that it is always better than

[1] https://www.robots.ox.ac.uk/~vgg/data/pets/

the baseline result where we predict all the pixels as belonging to the background instance. The less the downscaling of the image, the better the result. However, our implementation (and even existing implementations like the one of networkx) of the module that searches augmenting path is not optimal. Since our graph has cycles, finding an s-t path can lead to an interminable loop. Hence, our implementation of Ford-Fulkerson worked on 10×10 and 15×15 in very short execution time (few seconds) but the algorithm is very likely to be trapped in an interminable loop if we increase the size of the low dimensional image. One possible improvement to have more efficient search path module is to implement Kolmogorov version of Ford-Fulkerson [1] where the augmenting path is found from two non overlapping trees one contains the source while the other contains the tank. The two trees grow in different direction using non saturated edges until they meet on a node and build a s-t path.

In our project, we did not test this improved implementation, but preferred to try another family of algorithms rather than the augmentation path that will be detailed in the next section.

5.5. Push-Relabel

We have implemented Push-Relabel algorithm and tested it on some images from IIIT Pet dataset. We report some of our results in Figures 6-10. In those experiments, we progressively increase the size of the downscaled image on which the max-flow algorithm is applied. As shown in the first figure in 4, the bigger the downscaled image is, the better the result is. We have seen in our experiment that the mask gets more and more refined when the image is represented by bigger graphs as shown in the images of figures 6-10. Unlike Ford-Fulkerson, this algorithm allowed to test the algorithm on larger images in a finite execution time. It leads to very satisfying segmentation results with 30×30 downscaled images as shown in figure 10. The segmentation is accurate and can be improved with longer executions.

5.6. Scalability

In Figure 4, we reported the min-cut execution time for Push-Relabel algorithm for increasing image sizes (and thus increasing numbers of nodes and edges). Notice that even though we were able to improve the execution time over Ford-Fulkerson, it remains exponential with the number of nodes/trees. Therefore, we suggest to test other algorithms such as Kolmogorov in the future.

5.7. Impact of the seeds

In figure 11, we tried to test the effect of moving the unary connections around the object and the background. Notice that even if there is a small variations, the overall segmentation result is almost the same.

5.8. Comparison

Push-Relabel algorithm allowed to bypass the bottleneck part of Ford-Fulkerson algorithm that requires an efficient implementation of path search module (which is difficult to achieve even with libraries like networkx). According to the theoretical complexity, Ford-Fulkerson is more appropriate to image segmentation task. However, without an efficient implementation, Push-Relabel algorithm proposed by Goldberg-Tarjan is proven to be better according to our numerical experiment conducted in this project.

Another perspective is to compare graphical methods to deep learning frameworks. Our pipeline does not require any parametric model to be trained or tuned. It can be run on any image by simply transferring prior knowledge about the instance of certain pixels. However, deep models require training on a large dataset and the images on which inference is performed are limited to the set of images similar to the training set. Otherwise, the model is likely to suffer from generalization problems. Conversely, our pipeline is not exposed to generalization risks since it only depends on the appropriateness of the graphical representation of the image.

6. Code

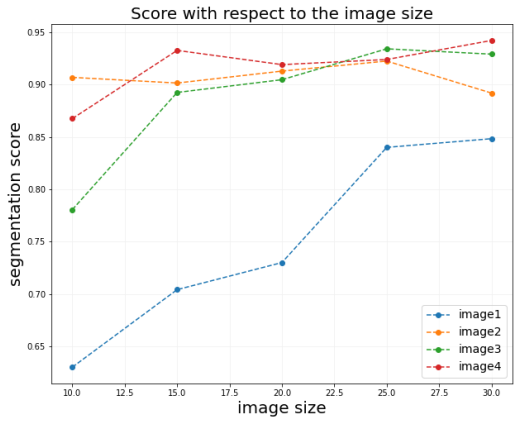
Our code is available in this github repository: `https://github.com/MarahGamdou/GRM-image-segmentation`

7. Conclusions

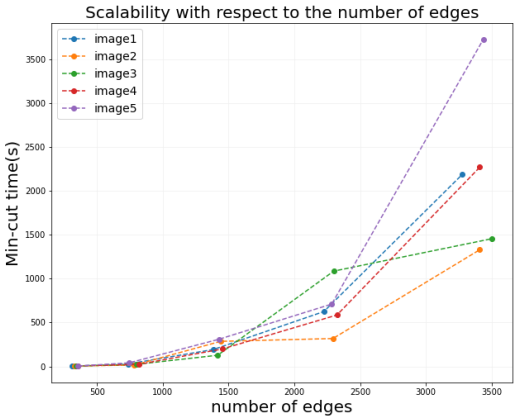
In this project, we tested two s-t min-cut algorithms for background/foreground segmentation task using a publically available dataset. We assessed the segmentation performance of each algorithm and compared their efficiency. Even if Push-Relabel was a noteworthy improvement of Ford-Fulkerson, there is still room for improvement in the execution time that can be fulfilled by investigating other techniques.

References

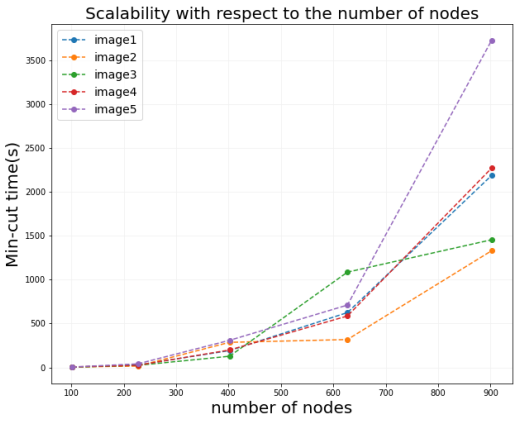
[1] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison ofmin-cut/max-flow algorithms forenergy minimization in vision. 2004. 4



(a) Pixel Accuracy score with respect to the image size

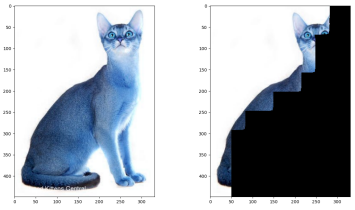


(b) Scalability with respect to the number of edges

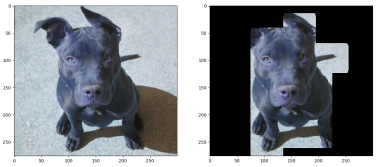


(c) Scalability with respect to the number of nodes

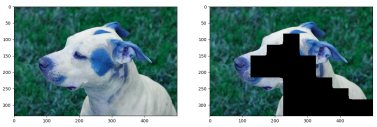
Figure 4: Push-Relabel Scalability



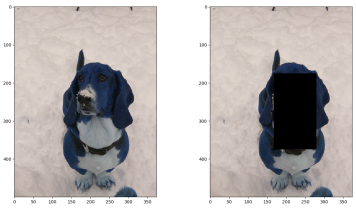
(a) 10x10, score = 0.70, time=0.2s



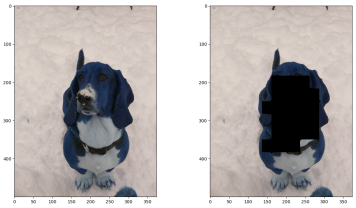
(b) 10x10, score =0.82, time=12s



(c) 10x10, score=0.87, time=5.5s



(d) 10x10, score=0.91, time=3.8s



(e) 15x15, score=0.91, time=2.2s

Figure 5: Ford-Fulkerson: segmented images

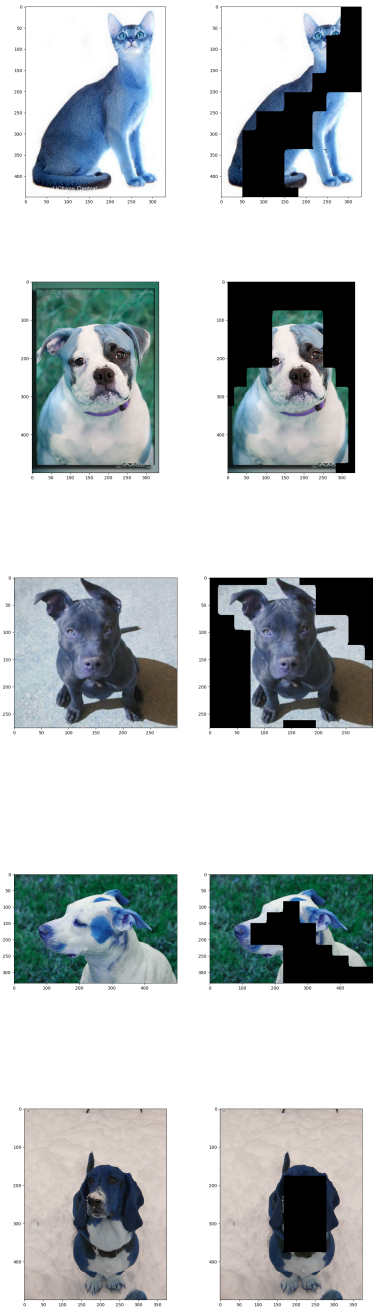


Figure 6: Push-Relabel: 10x10 images

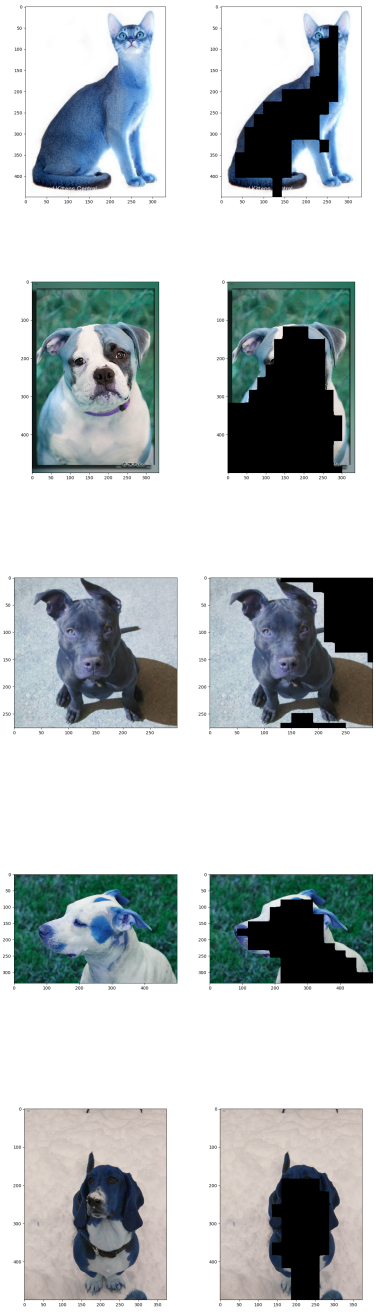


Figure 7: Push-Relabel: 15x15 images

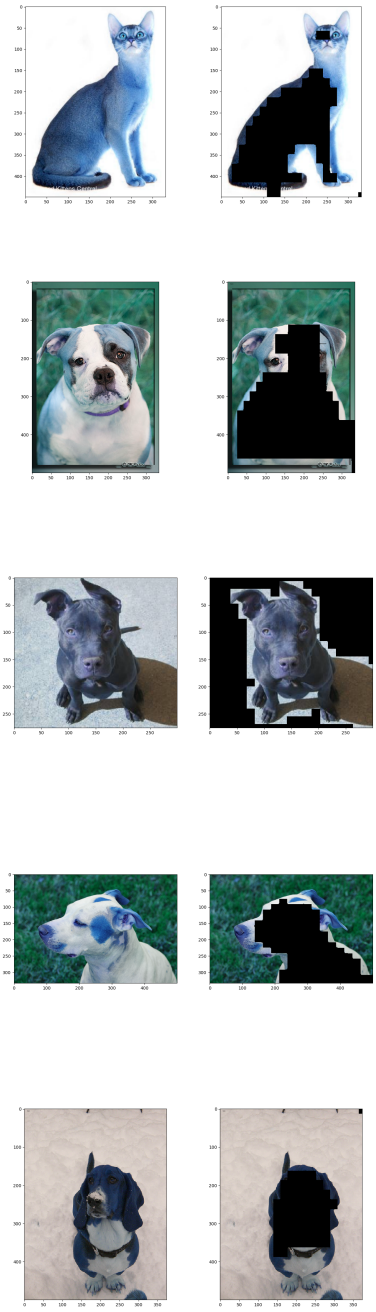


Figure 8: Push-Relabel: 20x20 images

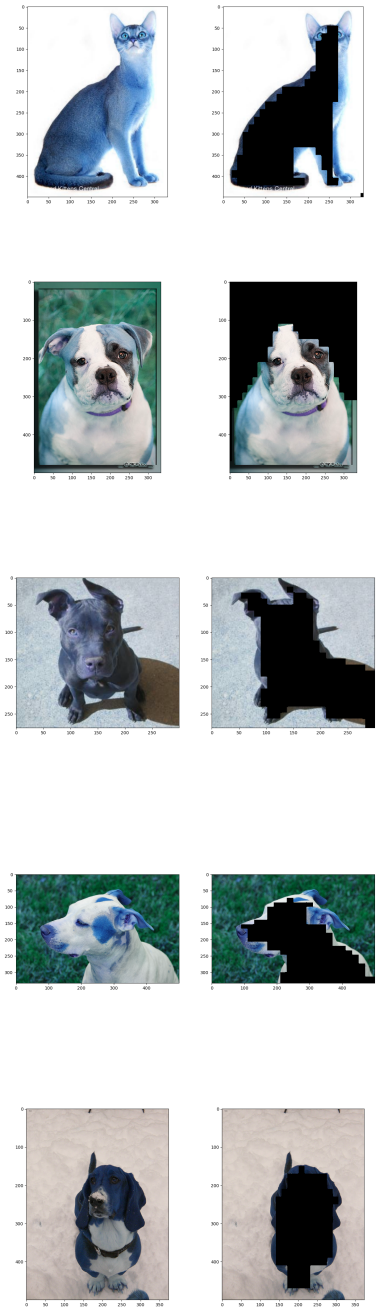


Figure 9: Push-Relabel: 25x25 images

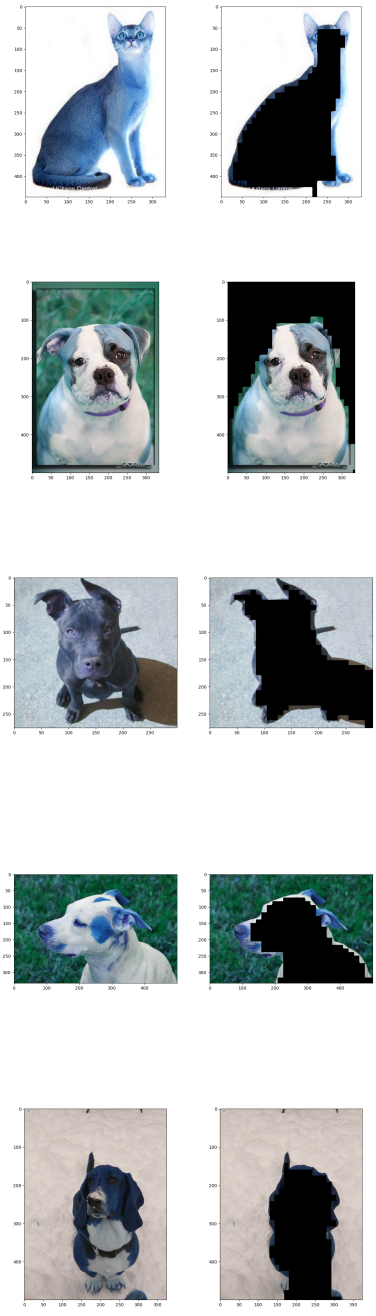


Figure 10: Push-Relabel: 30x30 images

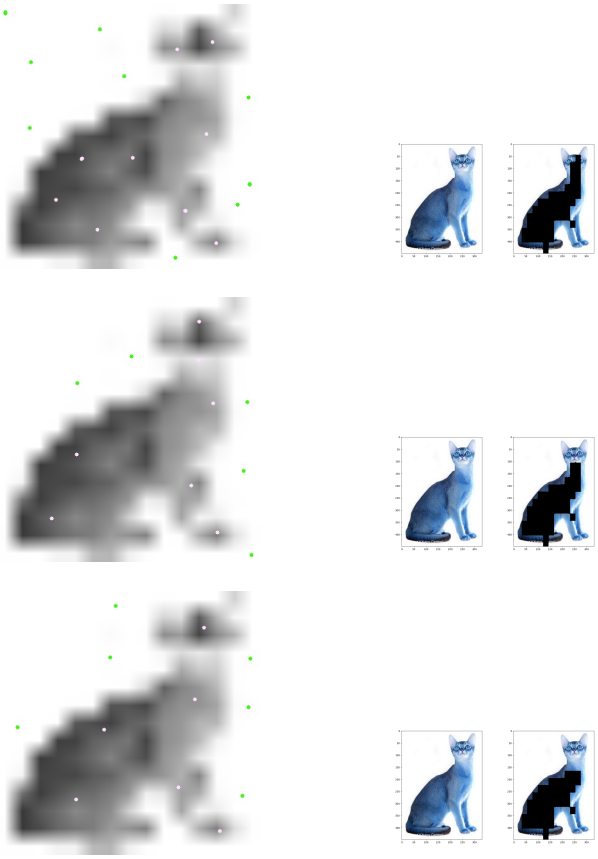


Figure 11: Moving the unary connections