Randomized Prim's Algorithm for Object Proposals

Li-Yun Wang
CS 506 (Project)
Spring Term, 2016

1. Introduction

Object discovery or generic object detection is similar with object detection, but the idea of generic object detection is to discover all of possible objects in the image. For instance, the generic object detection method tries to find a dog and a cat if given an image with two objects: a dog and a cat. In order to discover all of possible objects from an image, the idea of object proposals is proposed to address generic object detection application. Object proposals means that a bunch of bounding boxes is generated by object proposals methods when doing generic object detection. Because object proposals methods are object independent to object categories, these bounding boxes generated by object proposal methods will cover different object categories in the image.

The main work in this project is to discover all possible important objects when given a testing image. In other words, we would like to find useful objects, such as dog walkers, dogs, or leash, before doing visual situation recognition process. Thus, the purpose of this project is applying object proposals ideas to find a certain object from the image and diminish whole computation time in visual situation recognition process.

2. Hypothesis for Object Proposals

According to the paper proposed by Manen [1], they claimed that their proposed method is able to yield a bunch of bounding boxes within few seconds. In order to verify paper's

claims, two questions I would like to address are method's efficiency and detection rate performance for paper's method. The following statements will mention two questions more clearly.

Method's efficiency means that the object proposal method can find certain objects with few computation costs, which means the method is able to automatically output some useful bounding boxes after executing object proposal method. Useful bounding boxes represent that these bounding boxes cover important objects in the image. A fast method of finding important objects is benefit to visual situation recognition process.

Another question is method's performance given requested window numbers. That is, the hypothesis is that paper's method is supposed to detect most of objects in the image given a large requested window number. Since these bounding boxes include all possible objects, we would like to test whether the method can find three important objects, dog-walker, dog, and leash, in visual situation recognition or not.

3. Prior Work

Before verifying our hypothesis, randomized prim's algorithm and papers, which are related to randomized prim's algorithm, should be mentioned clearly. The first section is super-pixels extraction from and image. The second section is about standard prim's algorithm, and the next section mentions randomized prim's algorithm. The final section is to summarize randomized prim's algorithm.

3.1 Super-pixels Extraction

Super-pixels extraction means that image pixels should be grouped into a pixel's set according to evaluation criterions. In randomized prim's algorithm they applied effective

graph-based image segmentation method [2] to covert an image to a super-pixel set. The main idea in graph-based image segmentation is to measure the evidence for the boundary between two regions by using two quantities: external differences and internal differences. External difference is computing image's intensity difference between two different regions, so the intensity difference should be large when two regions or two pixels are different pixel categories. Internal difference is opposite to external difference and is based on intensity difference between neighboring pixels within the same region.

Thus, the first step in graph-based segmentation algorithm starts with an initial segmentation in the image, that is, each pixel represents a region. Then, the next step is to compute two measurements between regions for each node in a graph. The method merges two regions according to the measurement. If two neighboring regions are the same, these regions are merged together. After checking each node in the graph, the segmentation method generates a bunch of super-pixels for an image.

3.2 Prim's Algorithm

Prim's algorithm [3] is a tree-growing method by iterative procedure. The main idea of prim's algorithm is to find the shortest partial spanning tree from a weighted graph. Through applying this idea, prim's algorithm is useful on optimal problems in graph problems. More precisely, prim's algorithm finds a minimum spanning sub-tree or maximum spanning sub-tree in the graph, so the partial spanning tree can represent an optimal path in traveling problem or a region in the image.

Prim's algorithm includes several steps when building a partial spanning tree. First of all, the algorithm coverts a graph problem to a distance table, and each entry is the edge weight between two vertices. The next step is iteratively to pick up a next vertex from

neighbors of the current vertex according to the edge weight, so prim's algorithm will generate a minimum or maximum spanning tree dependent on different criterions.

3.3 Randomized Prim's Algorithm

Randomized prim's (RP) algorithm is similar with standard prim's algorithm to yield a partial spanning tree by iterative procedure. In general, randomized prim's algorithm is a graph-based method to generate a partial spanning tree from a weighted graph. Instead of modeling a spanning tree directly, RP algorithm incorporates two novel ideas into standard prim's algorithm. These two ideas are randomized ideas and learning method for setting edge weights in the graph. Randomized ideas mean that RP algorithm applies randomized mechanism to increase window's diversity and generate a scalar as the stopping value in each running process, which means the final spanning tree is different if running RP algorithm multiple times.

Another idea in RP algorithm is using learning method to find proper parameters in learning model and set up edge weights by well-trained parameters. In learning method logistic regression model is used to learn the relationship between two vertices given a bunch of training images. The purpose of using logistic regression model is that the negative logarithm likelihood function of logistic probability model belongs to concave function, which means final parameters must be a unique optimal value after training stage. For this reason, logistic regression model guarantees to find global optimal parameters given different initial parameters.

When finding proper parameters in training process, maximum likelihood method is a straightforward way to get proper parameters from initial parameters. The following equation represents maximum likelihood method in logistic regression.

$$\{\mathbf{w}^*, b^*\} = \underset{w,b}{\operatorname{argmax}} \sum_i y_i \ln \rho_i + (1 - y_i) \ln(1 - \rho_i) \quad \text{... (1)}$$

The idea of this equation consists of two possibilities represented by log-logistic function and utilizes this equation to find final parameters. Since the logarithm of likelihood function is a concave function, it is simple to use gradient descent method to iteratively update parameters in log-likelihood function. When computing logistic function, there are three quantities, color similarity between nodes, common border ratio, and the size to be three important feature values.

3.4 Randomized Prim's Algorithm for Object Proposals

Another issue is how to integrate randomized prim's algorithm to object proposals. So, basically, the first step is to get super-pixel groups by super-pixel extraction stage. After super-pixel groups generated, a weighted graph is also generated by super-pixel groups and well-trained parameters from learning stage. In the weighted graph a node represents a super-pixel, and the edge weight is calculate by logistic function. Finally, a weighted graph as the input to feed into randomized prim's algorithm. Randomized prim's algorithm is going to output a partial spanning tree.

Due to hypothesis in the previous section, we are able to assume that randomized prim's algorithm can achieve our hypothesis we made before. Two ideas in RP algorithm can support our hypothesis. One is that RP algorithm revises existing binary searching tree data structure to increase whole computation speed. This algorithm generates a bunch of bounding boxes within few minutes. Another idea is randomized ideas. Because of randomized ideas, RP algorithm generates different bounding boxes in each round. Applying these bounding boxes, RP algorithm is supposed to discover most of objects in the image.

4. Experiment Methodology

In order to verify our hypothesis for randomized prim's algorithm, different experiment designs should be proposed in this section. In the first and the second experiment we focus on results of detection rate to different objects given different factors: different intersection over union (IoU) value and different requested window numbers. The remaining experiments focuses on detected images, which include three objects by different measurement plots: cumulative plot and regular plot, given different windows.

4.1 Detection Rate of Intersection over Union Value

The first experiment methodology is to measure detection rate to different interesting objects, dog walker, dog, and leash, given different IoU values. The purpose of this experiment wants to validate the performance of RP algorithm to different objects because we are interested to certain objects in certain applications.

```
// A testing algorithm for different IoU values
Function IOU_Testing( Testing data, Label data, Parameters for RP algorithm )
{
   // initialize IOU array and object array
   1. IOU_array = [0.5, 0.6, 0.7, 0.8, 0.9];
   2. object_array = {'dog-walker', 'dog', 'leash'};

   // read ground truth file
   3. ground_truth_matrix = get_ground_truth( Label data, searching object );

   // object loop
    repeat{
      4. setting random seed

      repeat{    // image loop
         5. read one testing image
         6. run RP algorithm
         7. IOU_value = find_Proper_Window_Compute_IOU( Proposals, ground_truth )
      }until checking all of testing images
   }until checking all of elements in object array
   8. compute detection rate according to different IoU value
}
```

Generally speaking, the idea of testing RP algorithm is based on different IoU values and outputs detection rate for three objects: dog walker, dog, and leash given a testing dataset. For each object, this algorithm will look over all of images and compute IoU value for each testing image. After computing IoU value, the final step is computing detection rate

according to different IoU values. Thus, this algorithm generates detection rate for different IoU values. The following algorithm is a testing algorithm for different IoU values given a testing dataset.

4.2 Detection Rate of Different Requested Windows

The next experiment we would like to test is results of detection rate of different requested windows. Also, we would like to understand the variance in RP algorithm if repeating RP algorithm multiple times. The testing algorithm in this experiment also wants to verify detection rate to different objects, but instead of giving different IoU values, different requested windows are feed into this algorithm. In addition, this algorithm has to repeat RP algorithm several times.

The idea of this algorithm is similar with the previous testing algorithm. This algorithm needs to specify requested window array and get ground truth bounding boxes in the beginning of this function. The second stage is to compute results of detection rate on different requested window. During computing detection rate, the loop includes repeating loop to re-run RP algorithm five times and record detected numbers in each round. Final stage is to detection rate in each round and output final detection results. Whole testing algorithm is in the following algorithm.

```
// A testing algorithm for different requested windows
Function nWindow_Testing( Testing data, Label data, Parameters for RP algorithm )
{
   1. initialize requested window array

   // read ground truth file
   2. ground_truth_matrix = get_ground_truth( Label data, searching object );

   repeat{       // requested windows loop
      repeat{    // image loop
         3. read one image from testing data
         4. set requested window number

         repeat{      // repeating loop
            5. set random seed
            6. run RP algorithm
            7. IOU_value = find_Proper_Window_Compute_IOU( Proposals, ground_truth )
            8. if IOU_value >= 0.5, detected_number++;
         } until running five times
      }until checking all of testing images
      9. compute detection rate and store the result into result array
   }until checking all of requested windows in the array
}
```

4.3 Detected Number of Different Requested Windows

The purpose of the third experiment is similar with the second experiment, but this experiment verifies how many images, which includes three objects, given different requested window numbers when IoU value is equivalent or greater than 0.5.

The idea of the testing algorithm is also simple. The first stage is setting requested window array and dealing with ground truth data and images in the beginning of the algorithm. When testing different requested windows, a parallel computing tool is applied in the first small loop. The second small loop is to compute IoU values and record detection results, and the algorithm will save detection record as a file. When computing detected images, we will load three records for three objects and calculate detected numbers if three record results are equal to one.

```
// A testing algorithm for detected images given different requested windows
Function DetectedImages_nWindow( Testing data, Label data, Parameters for RP algorithm )
{
   1. initialize requested window array
   2. read ground truth file
   3. read images and store images into a structure array

   repeat{       // requested windows loop
      repeat{       // repeating loop
         4. set random seed
         repeat{ // image loop, apply parallel computing tool
            5. run RP algorithm
         } until look over all of images
         6. store proposal results
      } until running five times
      repeat{       // repeating loop
         repeat{ // image loop, apply parallel computing tool
            7. IOU_value = find_Proper_Window_Compute_IOU( Proposals, ground_truth )
            8. if IOU_value >= 0.5, records the detection result;
         } until look over all of images
      } until running five times
   }until checking all of requested windows in the array
   9. save detection records as file
}
```

4.4 Cumulative Number of Different Requested Windows

The final experiment focuses on cumulative number given different requested windows, which means we would like to use alternative perspective to examine RP algorithms' performance. Before running testing algorithm a preprocessing stage is dividing testing

data into ten disjoint small testing sets, and each set has the same number of testing images, and the following algorithm is whole testing algorithm.

```
// A testing algorithm for cumulative result
Function cumulative_function( Testing data, Label data, Parameters for RP algorithm )
{
   1. initialize requested window array and other variables
   2. read ground truth files for each object

   repeat{       // requested windows loop
      repeat{       // object loop
         3. set random seed
         repeat{ // image loop, apply parallel computing tool
            4. run RP algorithm
         } until look over all of images
         5. store proposal results
      } until look over all of objects

      repeat{ // image loop, apply parallel computing tool
         6. compute IoU value for three objects
         7. if IOU_value of each object >= 0.5, records detected numbers.
      } until look over all of images
      8. adjust the number of images after executing one requested window
      9. store detected information into structure array
   }until checking all of requested windows in the array
   10. save detection records as file
}
```

The algorithm includes two ideas: one is the usage of parallel computing tool and another idea is to adjust the number of testing images after executing one requested window. In order to diminish computation time when running a bunch of testing images many times, a parallel computation is used to compute final proposals in RP algorithm. Adjusting the number of images means that the algorithm only takes remaining testing images in the next requested windows. Thus, the number of total images in one set is decreasing when requested window is increasing.

5. Experiment

The experiment setting should be mentioned clearly before showing four experiments. The experiment platform is based on Linux machine, and all of testing scripts are written by Matlab2015a. In our testing image datasets we include two different datasets: Stanford testing data and Portland-State testing data. Each image in both of datasets has one dog walker, one dog, and one leash object.

5.1 Results for Detection Rate of Different IoU Value

In the first experiment we would like to know detection rate of different IoU values to different objects given two datasets. Figure one shows results for both of testing datasets. In each testing data, three objects are verified by RP algorithm separately. Picture's X-axis means different IoU values from 0.5 to 0.9, and Y-axis represents detection rate. In both of results, detection rate is decreasing if we increase IoU threshold. Compared to Stanford testing data, overall detection rate in Portland-State testing data is worse. There are two possible reasons to cause low detection rate. One is the ratio of objects over whole image is small, and another reason is randomized ideas in RP algorithm. When generating an initial node, RP algorithm randomly generates this node from a graph. If target object is small, target objects' super-pixel is difficult to be chosen by RP algorithm.

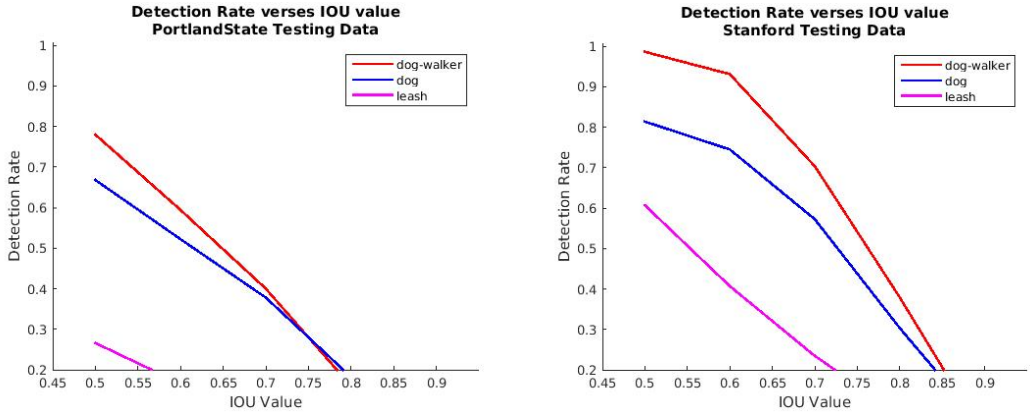

Figure 1, Detection rate results for different intersection over union value for both of testing datasets. Left picture is results of Portland-State testing data, and right picture is results of Stanford testing data. Each picture includes three results to three objects: dog-walker, dog, and leash.

5.2 Results for Detection Rate for Different Requested Windows

The next experiment focuses on results of detection rate given different requested windows for two datasets. In addition, experiment result includes the variance if running RP algorithm multiple times. Figure two is results for different objects given different testing data. Vertical lines represent the variance of repeating RP algorithms. According to results, we observe that detection rate is increasing if requested window number is also

increasing. RP algorithm is going to generate many bounding boxes when requested window is large. Because of this reason, RP algorithm has high detection rate compared to few requested windows.

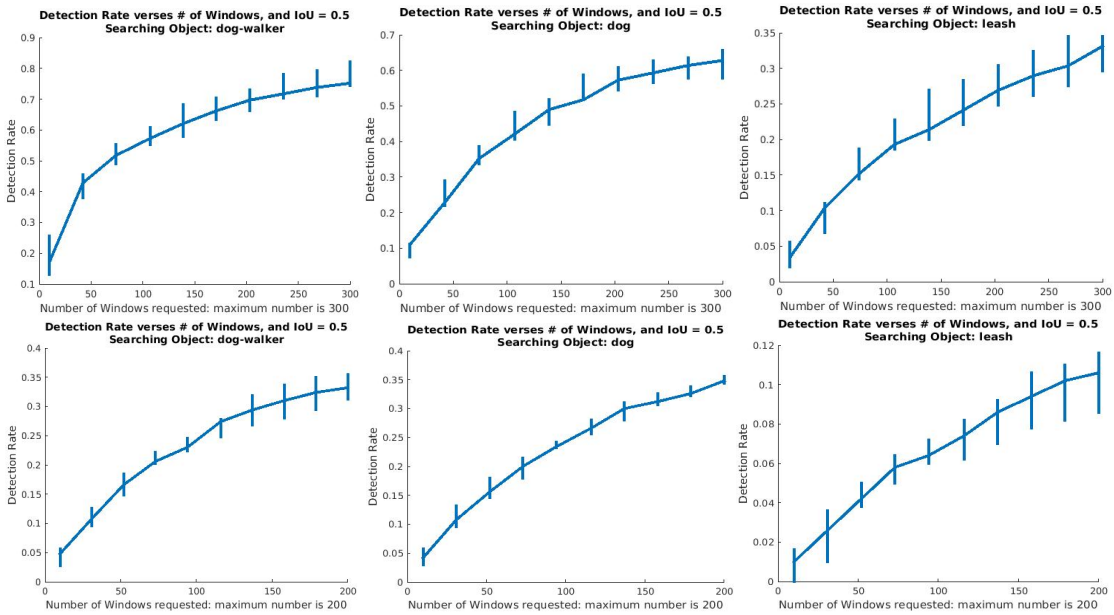

Figure 2, Detection rate for three objects given two datasets. The first raw is results of Stanford testing data, and the second raw is results of Portland-State testing data. Each vertical line is the variance of repeating RP algorithm five times at different requested window number.

5.3 Results for Detected Images for Different Requested Windows

The third experiment is similar with the previous experiment, but we are interested in detected images' performance given different requested windows. Figure three shows experiment results for both of datasets. In two plots overall detected image number is increasing if requested window number is increasing. But, if we compare Portland-State testing data with Stanford testing data, Portland-State testing data is better than Stanford testing data when maximum requested window is equal to eight hundred windows.

Another interesting point in this experiment is the variance result in two datasets. When maximum requested window is small, Stanford testing data has smaller variance than Portland-State testing data. In contrast, Portland-State testing data has small variance if

maximum requested window is greater than eight hundred windows. Because three objects are big in Stanford testing data, there are many overlapping windows in Stanford testing data when requested window is large. Many overlapping windows mean that many bounding boxes are associated with a target object, so the variance is big in Stanford testing data.

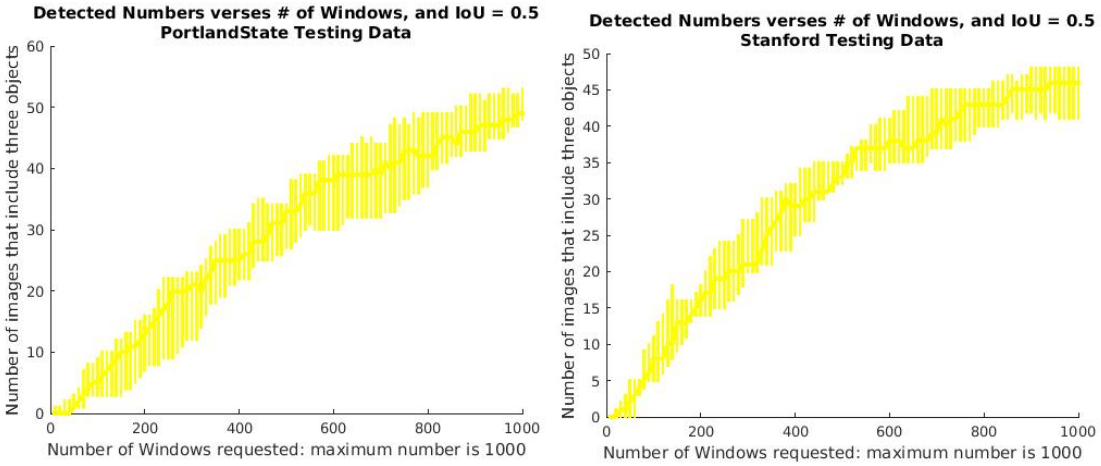

Figure 3, Detected images which include three objects given different windows. Left plot is the result of Portland-State testing data, and right plot is the result of Stanford testing data. Vertical lines mean the variance of repeating RP algorithm five times.

5.4 Cumulative Result for Different Requested Windows

The final experiment is cumulative image numbers given different requested window, and green-cross makers in this experiment represent cumulative results of running ten disjoint sets in a testing dataset. Since the plot represents cumulative results, detected number at each requested window is a cumulative value. For instance, detected image number at requested window twenty is equivalent to total number in requested window ten and twenty. In figure four detected image number is slightly increasing if requested window is also increasing, but overall performance is awful even maximum requested window is equal to five thousand windows.

The reason of low detected image is that RP algorithm is doing badly on leash object. In other words, this algorithm might detect dog-walker and dog in one image, but RP algorithm is difficult to discover leash object from the image. Thus, the image does not be taken into account of detected image because the algorithm only detects two objects from the image.

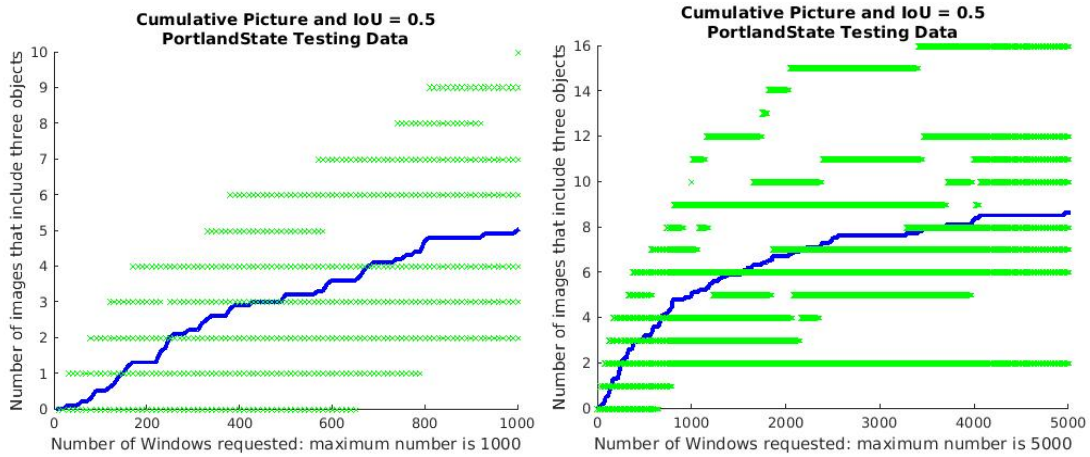

Figure 4, Cumulative detected images for different requested windows. Green markers represent cumulative detected images of running ten testing sets in Portland-State testing data at different requested window. Blue curve cumulative images of averaging ten cumulative results at different requested window.

6. Summary and Future Work

Randomized prim's algorithm is truly a stable algorithm for object proposals after we verified this algorithm by doing different experiments. Of course, RP algorithm is an efficient method for object proposals, which means RP algorithm is able to generate a bunch of bounding boxes within few seconds. The second advantage is method's performance. RP algorithm can discover most of important objects from an image if requested proposal number is large enough, so this algorithm has stable performance for most target objects. Final advantage is that RP algorithm can generates tight bounding boxes for each detected object. Due to tight bounding boxes, detected regions are useful

to be input images in other applications, such as ground truth generation and image recognition applications.

However, a big limitation in RP algorithm is dependent on object's size and object's type in the image. If target objects are small compared to whole image or belongs to non-rigid objects, RP algorithm is difficult to discover these target objects. For instance, in our testing data, the ratio of an area of leash object over whole image is small compared to other target objects: dog-walker and dog. In addition leash object has large variety of the contour, so RP algorithm has high failure probability to choose a super-pixel in leash object as an initial super-pixel node.

In order to solve an issue in RP algorithm, there are two possible solutions to solve the issue. One possible solution is integrating other detection techniques with RP algorithm to improve overall detection rate by setting few requested windows. Another possible solution is applying different learning method to build a weighted graph. Also, applying other powerful features may increase overall performance.

Reference

[1] S. Manen, M. Guillaumin, and L.V. Gool. Prime Object Proposals with Randomized Prim's Algorithm. In *ICCV*, 2013.

[2] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Computer Vision*, 2004.

[3] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 1957