Region-Maps

B. R. Siva Chandra

Abstract

Region extraction is a common pre-processing step in many computer vision and image processing tasks. The extrated regions are further analysed or processed for extraction of finer detail. This calls for efficient data structures for the regions extracted which should facilitate fast queries on these extracted regions. In this paper I propose a data structure called ***Region-Map*** which can facilitate such fast operations. This concept of region maps has been used extensively in the library ImprolaLib [1].

Keywords: C++, Class, Object, Data-structure, Time complexity, Linear search

1. Introduction

In most problems of image analysis, especially in bio-medical applications, we need to extract regions of interest from captured images. These regions are then, further analysed for finer detail. For such analysis, we typically would need to perform fast queries on the regions extracted. Examples of such queries include

- How many different regions have been identified?
- To which region does a given pixel belong to? Does it belong to any region at all?
- What is the size of that particular region to which a given pixel belongs to?

There are a number of cases where a large number of regions are extracted using the standard region extraction algorithms. Hence, in such cases an obvious query would be to know the number of regions extracted. This number can be used for further analysis, for example it could give the frequency of occurance of a particular artifact in the image. Also, this set of extracted regions will generally have falsely detected regions. This false detection occurs either due to the im-perfectness of the region extraction algorithm or due to human error. Such false detections would ideally be eliminated using an interactive tool where the user would click to indicate a false detection. This would require the second query of the list above. The system should be able to identify as to which region the pixel, on which the user clicked, belongs to. Typically the pixel count is used to quantify the size of a region. Also, the aggregate of the pixel counts of all the regions identified is used as a measure to quantify the total occurance of a particular artifact in an image. Hence an obviuos query would then be the third one from the above list.

In Section-2, I will describe the data-structure **Region-Map** in detail. In Section-3, I will explain the working of the region-map concept and also derive the time-complexities for the various queries. All description in this report is with respect to the implementation of the `RegionMap` class in the library ImprolaLib [1].

2. Region-Map: The Data-Structre

Figuratively or diagramatically, a region-map can be represented as shown in **Fig.1**. It is a 2-d array of pointers. Each cell of the array corresponds to the respective pixel of the image having the same indeces. Each cell of this array houses a pointer which points to a data-structure, Region.

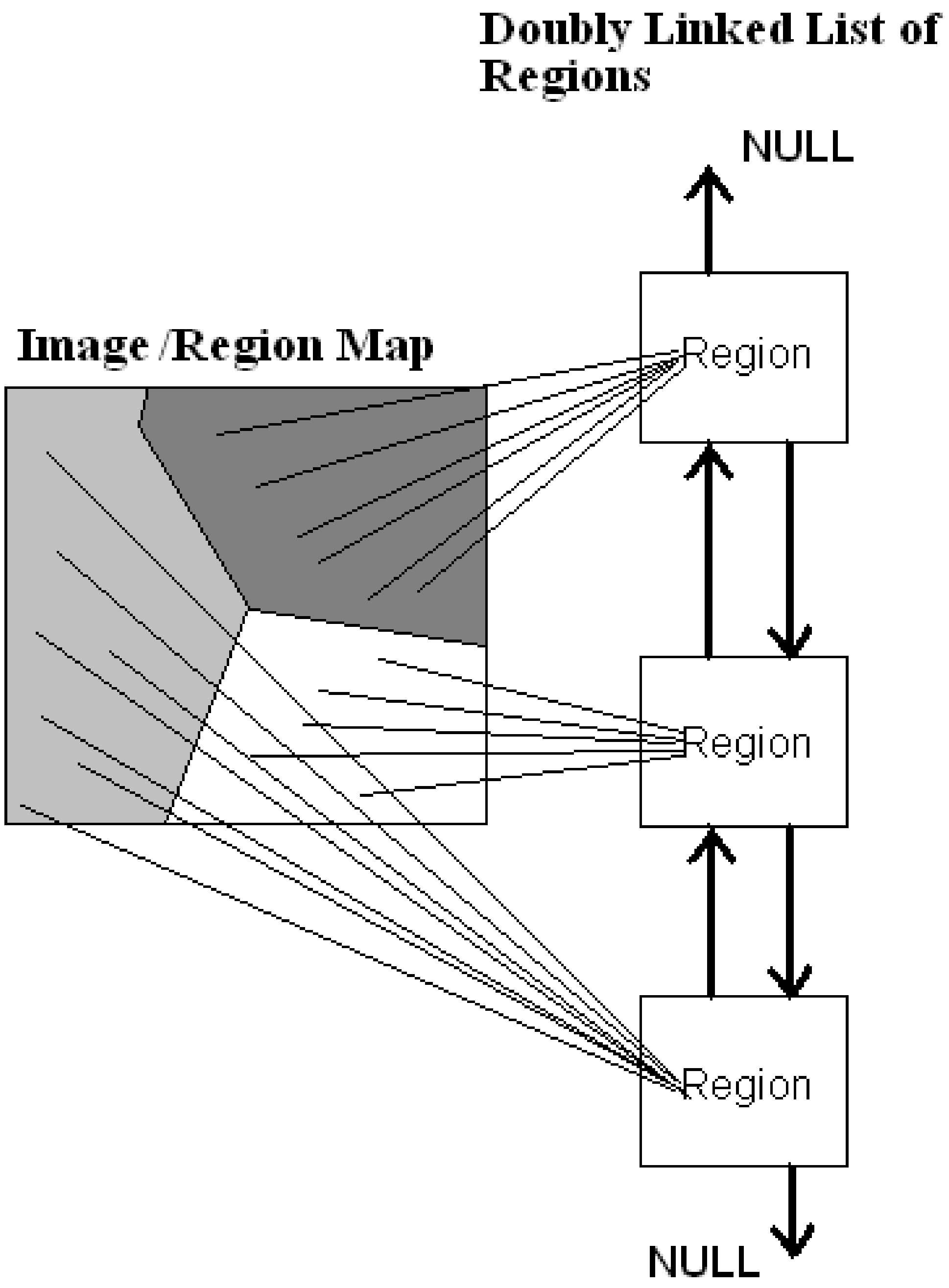

Fig.1 A Region-map.

Before going into what the above figure (**Fig.1**) conveys, let us look at other data-structures which go into the an efficient construction of a region-map. The basic data-structure is a **Point**. Its C++ definition is as follows.

```
class Point
{
    public:
        double x; // Stores the 'x' coordinate of the point.
        double y; // Stores the 'y' coordinate of the point.
};
```

The data-structure, Point, holds the ‘x’ and ‘y’ coordinates of any point. Using this, we build a data-structure, Region, which is a doubly-linked list of ‘Point’ objects. We again construct a doubly-linked list of such ‘Region’ objects to store our regions, which are extracted using a region extraction algorithm. This list is what is depicted in the right half of **Fig.1**.

Getting back to our data-structure of interest, the region-map, it is depicted in the left half of **Fig.1**. As mentioned earliar, it is a 2-d array of pointers. These pointers point to that region to which the corresponding pixel in the image belongs to. If a pixel in the image does not belong to any region, then the corresponding pointer of the region-map will point to `NULL`. The example in **Fig.1** shows an image with 3 different regions and each pixel within a given region point to its ‘Region’ object. At this point we should note that the list of ‘Region’ objects could instead have been a list of pointers to ‘Region’ objects. In that case, the pointers in the region-map will not point to ‘Region’ objects in the list but, the pointers in the region-map and the list will point to the ‘Region’ objects somewhere in memory space. This sort of approach is implemented in the `RegionExtractor` class of ImprolaLib [1].

3. Working of Region-Maps

In this section I shall show how a region-map can efficiently answer our queries listed in Section-1. Also, for each query, we shall look at the time complexity involved in arriving at an answer. The space complexity of a region-map is obviously of $O(MN)$, where $M \times N$ is the size of the image involved.

We should be maintaining a doubly-linked list of the regions extracted, either as a list of pointers to the ‘region’ objects, or as a list of ‘Region’ objects themselves. If the implementation of the list maintains an attribute to indicate its size (i.e the number of list nodes), then knowing the size of the list is an $O(1)$ process. Time required to determine the sizeof the list would be independent of the size of the list. If the implementation of the list does not maintain a size attribute, then determining the size of the list would be an $O(N)$ process where N is the size of the list. Hence we see that, to answer the first query in Section-1, the time complexity is independent of the underlying region-map. Similarly, since a ‘Region’ object is itself a doubly-linked list, the time required to know the size of a particular region would be independent of the underlying region-map. Hence, time required to answer the third query listed in the list in Section-1 is also independent of the underlying region-map.

The main advantage of using a region map is its usefullness is answering the second query of the list in Section-1. Given any pixel, it is an $O(1)$ process to find out the region to which it belongs to using the region-map of the extracted regions. This is possible because accessing an array element requires an amount of time which is independent of the size of the array. Also, in our case, it is independent of the number of regions-extracted. If we did not use a region-map for our regions, then finding out as to which region a given pixel belongs to would involve two nested linear searches whose time complexity depends on the number of regions extracted and on the size of the regions extracted.

References

[1] Siva Chandra. ImprolaLib. *http://improla.sourceforge.net/*.