

# AN ALGORITHM TO FIND THE AREA OF IMAGE OBJECTS USING POLYGONAL APPROXIMATION

M. Ashraful Amin<sup>1</sup>, Shamim Akhter<sup>2</sup>, Nitin V. Afzulpurkar<sup>3</sup>

<sup>1</sup>Department of Computer Science and Information Management, Asian Institute of Technology, Thailand.

<sup>2</sup>Department of Computer Science, American International University of Bangladesh, Bangladesh.

<sup>3</sup>Department of Industrial System Engineering, Asian Institute of Technology, Thailand.

E-mail: amin021us@yahoo.com, shamim@aiub.edu, nitin@ait.ac.th

## ABSTRACT

One of the important issues in the image feature is the area of objects in the image. Here we propose a method to calculate the area of an image object. This method finds the polygonal area approximation of image objects. A sequential fine-tuning of this polygon is used to estimate the actual area. At first, we present an algorithm to estimate the object area from Convex-Hull approximation. Finally, a polygonal approximation is used to find the object's area to reduce the Convex-Hull calculation overhead.

## 1. INTRODUCTION

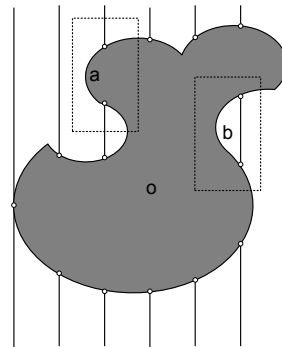
Different types of feature [1, 2] descriptions are in practice in image processing field. One of the common features is the 'Area' of image objects. Convex-Hull is an approximation of image area and feature. Here we propose an algorithm that will find more precise area of image objects from Convex-Hull approximation. Finally, a faster algorithm than the initial one is proposed. Where we discuss how it is possible to find the precise area of an object from an arbitrary polygonal approximation.

This algorithm is inspired by some common image processing algorithms such as priority boundary detection [5] and Convex-Hull [8]. There are several algorithms available for detecting the boundary of an image object. Some of them are shaped in the form of morphological operators [1], whereas others are based on boundary refining or boundary growing [5]. We will discuss two algorithms; concept of these algorithms is used in the proposed algorithms.

### 1.1 Boundary refining & boundary detection

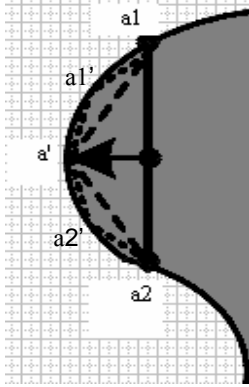
We focus on the approximate boundary estimating algorithms. And utilize these algorithms to find more precise area to contribute in feature

calculation. There are two algorithms that can be used consecutively to grow a better boundary of an image object. The First algorithm is to find the likely position of the boundary of the image object. The priority boundary detection algorithm will find points on the boundary in intervals (depending on some understanding of the concerned object). Figure-1 shows a situation where prior knowledge of the approximate center (o) of the object is used. If we lack information about the object in hand then we can scan from left to right and then find the other points of boundary in a fixed period or interval (Figure-1).

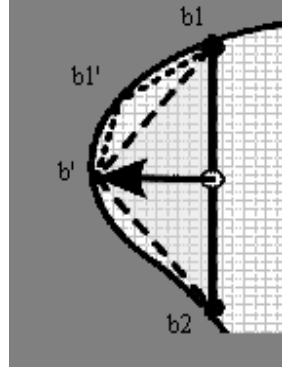


**Figure-1:** Equal spaced parallel lines to determine points of initial guess on the object boundary

Here we present two enlarged sections of the concerned object from Figure-1. Figure-2.a shows region 'a' where the first guess of boundary from the prior knowledge is denoted by 'a1' and 'a2'. Then we fine-tune this by the divide and conquer method. Each time we discover a point on the boundary by the perpendicular line from the midpoint of the guessed boundary line (here initial boundary is the a1a2 line Figure-2.a). Similar approach is followed for portion 'b' (Figure-2.b). The only difference is the search direction; one is outwards (Figure-2.a) other is inwards (Figure-2.b).



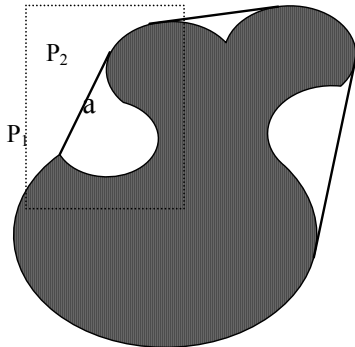
**Figure-2.a:** Enlarged region 'a' of object from Figure-1.



**Figure-2.b:** Enlarged region 'b' of object from Figure-1

## 1.2 Convex-Hull

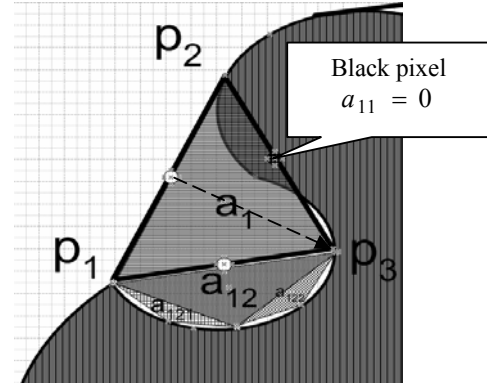
The Convex-Hull [8] feature of an object is the smallest convex polygon that encloses the entire object. As we associate the criteria “smallest polygon”, finding the points on the boundary to grow the Convex-Hull is not an easy task [8]. It might take substantial amount of time depending on the shape of the object in hand. Moreover, keeping the property of convexity is another task to be noticed. In Figure-3, we present an example of Convex-Hull. The area enclosed by the convex polygon is greater than or equal to the actual area of the object. In the development of our proposed algorithm, first we utilize the existing Convex-Hull algorithm to calculate more accurate area. Afterwards, a polygonal approximation is used to find the object's area to reduce the Convex-Hull calculation overhead.



**Figure-3:** Convex-Hull of the image object

## 2. REFINED AREA FROM CONVEX-HULL AREA

In the previous section we have briefly discussed the characteristics of Convex-Hull area. In this part we will assume that we know the Convex-Hull area



**Figure-4:** Using the refinement algorithm on the Convex-Hull (region 'a' from Figure-3 400% zoomed)

( $A_{convex}$ ) and the array of vertices ( $P = [p_1, p_2, p_3 \dots p_n, p_1]$ ) of the convex polygon that convey the area. Detail algorithm, that finds the convex polygon and area, can be found at [6]. We propose an algorithm (Figure-5) to find a more precise area of an object ( $A_{obj}$ ) from the convex polygon.

### 2.1 Algorithm:

```

object_area ( $A_{convex}, P[]$ )
{
   $A_{obj} = A_{convex}$ ;
  for ( $i=0; i < \text{sizeof}(P); i++$ )
  {
     $point\_1 = P[i]$ ;
     $point\_2 = P[i + 1]$ ;
    modify_area( $point\_1, point\_2$ );
  }
}

modify_area ( $point P_1, point P_2$ )
{
  if( $\text{mid\_point\_of\_line}(p_1, p_2) = \text{white\_pixel}$ )
     $p_3 = \text{mid\_on\_surface}(p_1, p_2)$ ;
  else  $p_3 = p_1$ 
     $A_{obj} = A_{obj} - \Delta p_1 p_2 p_3$ ;
    if ( $\Delta p_1 p_2 p_3 > \epsilon$ )
    {
      modify_area( $p_1, p_3$ );
      modify_area( $p_2, p_3$ );
    }
}

```

**Figure-5:** Algorithm to find object area from Convex-Hull

There are three major functions in this algorithm (Figure-5). The first function (*object\_area*) has convex area and the vertex array of the convex polygon as parameters. This function can be called for all the individual objects in the image. If there is a hole inside an individual object then the hole is considered as an individual object and using the similar procedure for Convex-Hull of the hole, the hole-area can be calculated and then subtracted from the actual object area. This can be followed in chain. The second function (*modify\_area*) is called recursively as we observe self-similarity in the scenario. This process is similar to the boundary smoothing using “divide and conquer” algorithm that we mentioned in the previous section. The function (*mid\_on\_surface*) takes two points ( $p_1, p_2$ ) and returns a point ( $p_3$ ) on the boundary of the object that is cut by the perpendicular line on the mid-point of the line  $p_1p_2$ .

There are two terminating conditions for the recursion. An active termination depends on the value of  $\mathcal{E}$  (the smallest area concerned). The passive termination condition is  $\Delta p_1 p_2 p_3 = 0$ .

This happens if the midpoint of  $p_1 p_2$  is a dark (object) pixel (Figure-4) and we forcefully make it by  $p_3 = p_1$ . In Figure- 4 we have presented a diagram that shows one iteration where unlike ( $p_2, p_3$ ), ( $p_1, p_3$ ) is farther processed because the midpoint is a white (background) pixel.

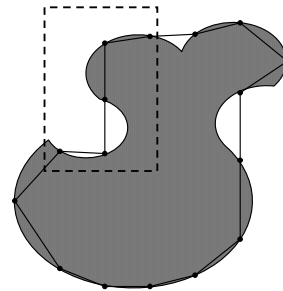
In Figure-4, notice that  $\Delta p_1 p_2 p_3$ , includes some area that is indeed part of the main object area. This is one problem with this proposed algorithm. The other problem with this algorithm is the overhead of calculating the Convex-Hull (both the area and the convex polygon). Next, we will modify this proposed algorithm to overcome these two problems.

### 3. REFINED AREA FROM ARBITRARY POLYGON

The overhead of calculating the convex can be reduced by using a polygon that roughly encloses the image object. As mentioned in the introduction the initial points on the boundary can be discovered based on a prior knowledge about the object, or we can scan from left to right through the image to get the first hit. In this algorithm we will simultaneously scan through the image from left to right and right to left. In this process, we will be able to discover the points  $p_1$  and  $p_9$  of the example image (Figure-6).

Then we will find the other points of the potential polygon by dividing the object into equal parts. Here we have divided the object into six equal parts (Figure-6).

Instead of convex polygon, to measure the area of the object we will get an arbitrary polygon which can be viewed as an array of vertices  $P = [p_1, p_2, p_3 \dots p_n, p_1]$  here  $p_1 = (x_1, y_1)$ ,  $p_2 = (x_2, y_2) \dots$  are points or coordinates in the image plane. The polygon  $P$  is triangulated and the area is calculated [8]. There are two types of errors in the area calculation of the image object using the polygonal approximation. The “positive error” occurs when we are not enclosing the area that is indeed part of the object. The “negative error” is the extra area that is not an actual object area but added in the approximation. Accurate object area is found by refining polygonal area ( $A_p$ ).

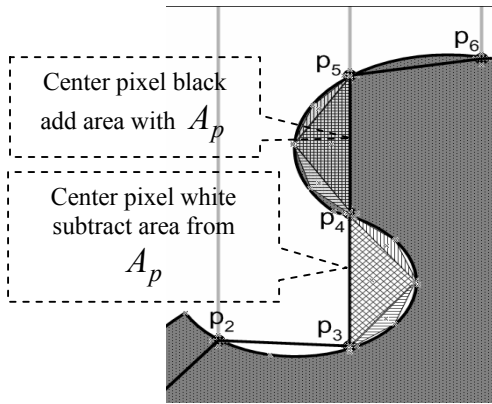


**Figure-6:** An arbitrary polygon for the image object

This process is quite similar to the process described in the previous section. The major difference is that in the previous algorithm, only the “negative errors” were our concerning factor, but this time there are also “positive errors”. The modification is trivial, in case of getting a white (background) pixel as the middle of two points we will subtract as usual and in case of black (object) middle pixel we will add (Figure-7).

Previously, we were subtracting the extra area that were included in the convex polygon, this time the extra area will not only be subtracted but also be added that is missed in the polygonal area approximation. Notice that because of this the passive termination condition is no more needed for the recursion, the active termination condition (the threshold  $\mathcal{E}$ ) is sufficient.

There are two major functions in Algorithm-3.1 (Figure-8). The first function (*object\_area*) has three parameters: an image, an integer that determines the smoothness of the approximating polygon and the third one is active stopping criteria



**Figure-7:** Positive error and negative error (dotted reason from Figure-6)

of the recursion. This function finds the polygon based on the **no\_of\_slice** from the image, finds area of the polygon and then calls the other function. The second function (**modify\_area**) is to add/subtract the miscalculated area. This function is similar to the

### 3.1 Algorithm:

```

object_area (image [m][n], no_of_slices,  $\epsilon$ )
    { index=0;
     $p_l = \text{left\_end\_point\_of\_image\_object}(\text{image});$ 
     $p_r = \text{right\_end\_point\_of\_image\_object}(\text{image});$ 

     $\text{slicing\_amount} = \frac{|p_r.x - p_l.x|}{\text{no\_of\_slice}};$ 
     $P = \text{vertices\_of\_polygon}(p_l, p_r, \text{slicing\_amount});$ 
     $P[\text{sizeof}(P)] = p_l$ 
     $A_p = \text{area\_of\_polygon}(P);$ 

     $A_{obj} = A_p;$ 
    for i from 0 to sizeof (P) - 1
        modify_area (P[i], P[i + 1])
    return area_feature;
    }
modify_area (point P1, point P2)
    {  $p_3 = \text{mid\_on\_surface}(p_1, p_2);$ 
    if (mid_point_of_line(p1, p2) = whitepixel)
         $A_{obj} = A_{obj} - \Delta p_1 p_2 p_3;$ 
    else
         $A_{obj} = A_{obj} + \Delta p_1 p_2 p_3;$ 
    if ( $\Delta p_1 p_2 p_3 > \epsilon$ )
        { modify_area(p1, p3);
        modify_area(p2, p3)
        }
    }

```

**Figure- 8:** Algorithm to find area from polygonal approximation

**modify\_area** of algorithm 2.1 except here the area modified not only by subtracting but also by adding. Thus the midpoint of the line  $p_1 p_2$  is checked. If the midpoint is a background (white) pixel then the triangle area is subtracted and if it is a black pixel (object) the triangle area is added.

## 4. CONCLUSION

Here we have presented a novel way to find the area of an image object. The calculated area can obviously be used in any image processing problems that require measurement of area. We were initially motivated by the refinement algorithms that are in practice in Image Processing mainly for boundary detection of objects in the image. Our proposed algorithm uses a similar approach like the boundary detection by boundary refinement. With the boundary refinement process at the same time we estimate the area of the object of which the boundary is detected. The idea is first we have assumed that the area of the Convex-Hull polygon of the object in hand is known, and we refine the convex polygons area and estimate the object area. We notice two problems in this algorithm. One is the overhead of calculating the convex polygon and the second problem is sometimes depending on the shape of the object we miscalculated area in the area modification process (Figure-4). Finally we proposed a better algorithm that reduces the overhead of calculating the Convex-Hull and the second problem is also resolved.

## REFERENCES

- [1] Russ, John C., *The Image processing Handbook*, 2<sup>nd</sup> Ed, CRC Press, 1995.
- [2] Jain, Anil K., *Fundamentals of Digital Image Processing*, Prentice-Hall of India Private Limited, New Delhi, India, 2003.
- [3] Gonzalez, Rafael C. and Woods, Richard E., *DIGITAL IMAGE PROCESSING*, Addison-Wesley Publishing Company, Inc., USA, 1993.
- [4] <http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Morpholo.html>
- [5] Awcock, G. J. and Thomas, R., *Applied Image Processing*, McGraw-Hill, Inc. Singapore, 1996.
- [6] <http://www.icaen.uiowa.edu/~dip/LECTURE/Shape3.html#convexhull>
- [7] <http://www.icaen.uiowa.edu/~dip/LECTURE/ImageProperties3.html#properties>
- [8] Cormen, T. H., Leiserson, C. E and Rivest, R. L., *INTRODUCTION TO ALGORITHMS*, Prentice-Hall of India Private Limited, New Delhi, India, 1998.
- [9] <http://mcraefamily.com/MathHelp/GeometryPolygonAreaDeterminant.htm>