

Maximum Ball Skeleton Object Recovery

CSE 573: CVIP - Project

PROBLEM STATEMENT

The objective is to design an algorithm to compute maximum ball skeletons of all objects in a given binary image, and label each skeleton pixel by the radius of its maximum ball. It has to be shown that reconstruction of the original image from its labeled skeletons by starting from a binary test image, constructing the labeled skeletons, and then from that reconstructing the original image exactly is possible. The algorithm should permit the use of any distance measure, 4-connected, 8-connected or Euclidean distance.

I. RELATED WORK

The Maximum Ball Skeleton, also called Medial Axis Transformation (MAT), was first proposed by H. Blum [1]. The skeleton of an object is the locus of points that are minimally equidistant from at least two points on the object's boundary. In a 2D image it is called the Medial axis, and in a 3D image it is called a medial surface [2]. Applications of Medial Axis Transformation are Shape matching, Animation, Dimension reduction, Solid modeling, sharpening of shape, Motion planning, Mesh generation [4]. 3D skeletons can be used to perform geometric operations like proximity calculations, shape decomposition [2]. A related algorithm, Hilditch's Algorithm for Skeletonization works inwards from the boundary and removes all the pixels except the skeleton pixels [3] and [4].

II. PROPOSED SOLUTION

We chose to divide this problem into two components; Skeletonization and Image Reconstruction. The skeletonization component deals with constructing the maximum ball skeleton for a given image consisting of one or more connected blobs. The image reconstruction component deals with reconstructing the original image, given its maximum ball skeleton. We later added a third component, Image Reconstruction Validation, to check how close the reconstructed image was to the original image.

We chose not to perform segmentation on the image, to extract the various connected blobs for the following reason. We deal with only the foreground pixels in an image. We think of the process of skeletonization as a process of covering, with a blanket, each of the islands in a collection of islands. The moment we touch the sea (background pixels) we stop. Even in the extreme case where two blobs are separated by just one background pixel, the algorithm works fine. We show this is case in the tests that we have conducted and documented.

Skeletonization Component

Recall that the skeleton of an object is used to represent the object in a compact form. Using the skeleton we should be able to reconstruct the original image exactly. In other words, the maximum ball skeleton reconstruction technique should provide a lossless compression method.

We chose to use the brute force method of computing the maximum ball skeleton for an image. In this approach, we raster scan the image for different ball sizes r , starting at 1 to a maximum of K , where $K = \min(\text{Number_of_rows_in_Image}/2, \text{Number_of_columns_in_Image}/2)$, and check if we can construct a ball B of radius r . If we can, then we mark the current pixel as a skeleton pixel and do the following test for each pixel in the ball B . Test whether the maximum ball of size $\leq r-1$ centered at this pixel is completely contained in the ball B . If the condition holds then we relabel that pixel as a non-skeleton pixel. This implementation is intuitive and easy to follow. We worked on the assumptions that, we only had to find the maximum ball skeleton for binary images and an image could have any number of foreground blobs.

Pseudo Code for Skeletonization

```

Set  $r = 1$ 
Loop While TRUE
  Raster scan through the image
  Loop for each image foreground pixel  $p_{ij}$ 
    Check if we can construct a ball  $B$  of size  $r$  centred
    at  $p_{ij}$ 
    If TRUE
      Re-label  $p_{ij}$  as a skeleton pixel with quench
      function  $q(p_{ij}) = r$ 
      Loop for each skeleton pixel  $np_{ij} \in B$ 
        Check if the existing maximum ball, centered
        at pixel  $np_{ij}$  is completely contained in  $B$ 
        If TRUE
          Re-label  $np_{ij}$  as a non-skeleton pixel
        End If
      End If
    End Loop
  Increment  $r$  by 1
  If cannot centre a ball of size  $r$  at any pixel  $p_{ij}$ , for all
  pixels in the image, stop skeletonization and return
  skeleton
End Loop

```

Maximum Ball Skeleton Representation

We thought up of two methods for representing the maximum ball skeleton that was computed for an image. The first method stores the skeleton in a sparse matrix that is the size of the image canvas itself. Each pixel in the matrix, that corresponds to a skeleton pixel, is given a value that is equal to its quench function $q(p_{ij})$. We decided on a second method of representation when we realized that the skeletons for most images we tested were sparse matrices with a considerably higher number of non-skeleton pixels than there were skeleton pixels. This prompted us to consider storing just the skeleton pixels in a separate data structure. This data structure was easily realized by using a $(n+1)$ -by-2 array of values, where n is the number of skeleton pixels. The first and second columns in row i of the array, stores the row and column indices for the i^{th} skeleton pixel. The third column of the i^{th} row stores the value of the quench function $q(p_{ij})$ for the i^{th} skeleton pixel. The array has one extra column which is used as a location to store the size of the image. This row stores the row, column and pixel value of the bottom right pixel in the skeleton image. Using these values, we know the size of the image. We give the user an option to use either of these data representations. We thought that using the second representation was more meaningful since the objective of skeletonization is compression. With the first method, the absolute compression ratio, i.e. *size_of_original_image / size_of_skeleton*, is 1 since we are storing the non-skeleton pixels in addition to the skeleton pixels. The second method achieves lower compression ratios and the lower the ratio, the more efficient is the compression technique.

Image Reconstruction Component

The pseudo code for image reconstruction is fairly straightforward. We simply take every maximum ball skeleton pixel and grow a ball of size S around that pixel, where S is the value of the quench function $q(p_{ij})$ associated with that skeleton pixel. We have to be careful in using the correct internal representation while reconstructing the image. If we are storing the maximum ball skeleton in its compressed representation, we need to convert this to the sparse matrix form before reconstruction can be performed. This conversion is done by reading off the row and column index values for each skeleton pixel entry and then setting the value of the pixel location indexed by these values to the value of the quench function $q(p_i)$ associated with the i^{th} row of the compressed skeleton representation.

Image Reconstruction Validation Component

The maximum ball skeletonization technique is proved to be a lossless compression method since the location of the skeleton pixels and its associated quench function values form an invertible transform. Using just these values, it is possible to reconstruct the image that the skeleton represents, exactly. We decided to test this and test the correctness of our implementation by validating the reconstruction of the image against the

original image. There are two types of errors that can occur when we reconstruct an image given its maximum ball skeleton. Firstly, some pixels that were present in the original image could be absent in the reconstructed image. This means that the skeleton has not been able to capture all the information that was present in the original image. This error can be detected by raster scanning both the original as well as the reconstructed images, and counting the number of times the reconstructed image shows a background pixel when the corresponding pixel in the original image shows a foreground pixel. Secondly, spurious pixels, that were not present in the original image, could get added to the reconstructed image. This means that the skeleton is adding noise to the reconstruction. This error can be detected by raster scanning both the original as well as the reconstructed images, and counting the number of times the reconstructed image shows a background pixel when the corresponding pixel in the original image shows a foreground pixel. These two types of errors combine to give the overall inaccuracy in the reconstruction of the image from the skeleton.

III. TESTING AND ANALYSIS

We first present a few images which depict what the maximum balls for a given quench function value and distance metric are supposed to look like.

Figure 1

Figure 1 shows a set of skeleton pixels having quench function values 0, 2, 4 and 9 respectively from left to right.

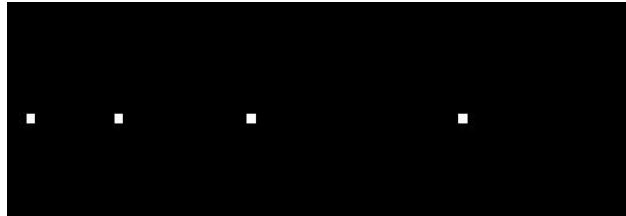


Figure 2

Figure 2 shows the 4-connected maximum balls that can be grown for each skeleton pixel in figure 1 above, using its associated quench function.

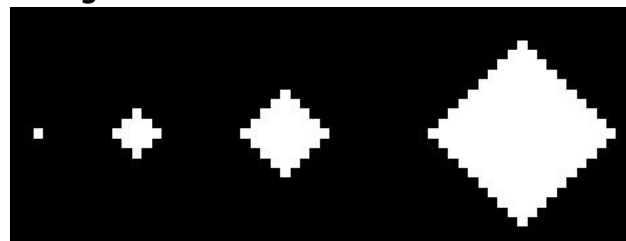


Figure 3

Figure 3 shows the 8-connected maximum balls that can be grown for each skeleton pixel in figure 1 above, using its associated quench function.

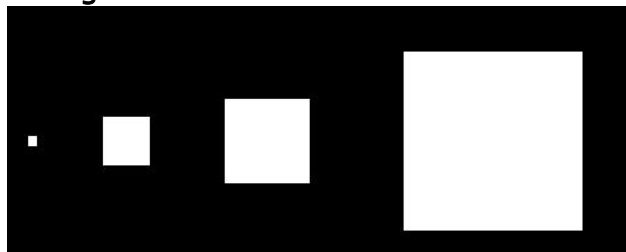
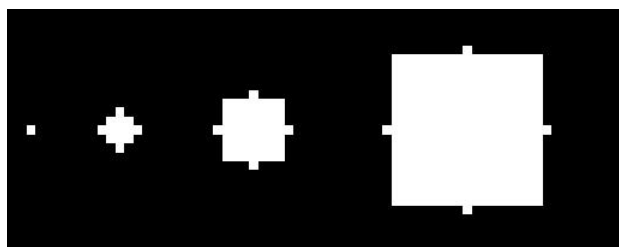


Figure 4

Figure 4 shows the Euclidean maximum balls that can be grown for each skeleton pixel in figure 1 above, using its associated quench function.



We ran the skeletonization algorithm on a set of images that were chosen to test the correctness of the implementation. We now describe the set of images we chose and the rationale behind choosing them to test the algorithm as well as a description of the results. All the figures, shown below, have three images each. The first is the original image whose skeleton we wish to compute. The second is the computed maximum ball skeleton of the image and the third is a reconstruction of the image using the maximum ball skeleton. For all test cases we ran, we achieved a 100% image reconstruction, i.e. no extra spurious pixels were added and neither were any original pixels, removed in the reconstruction. Note that all the images below are logical (binary) images. Regions in white represent foreground pixels and regions in black represent background.

Arbitrary Image

We used the 'plane.gif' image found in the database of images on the CSE-573 course website <http://www.cse.buffalo.edu/faculty/peter/cse573/dbs/>, for this test. This image represented an arbitrary image as seen in Figure 5. It consists of just one connected blob. It shows some symmetry about the vertical axis but is not perfectly symmetric. This is reflected in the roughly symmetric skeleton produced. The distance measure used for this test run was 4-connected.

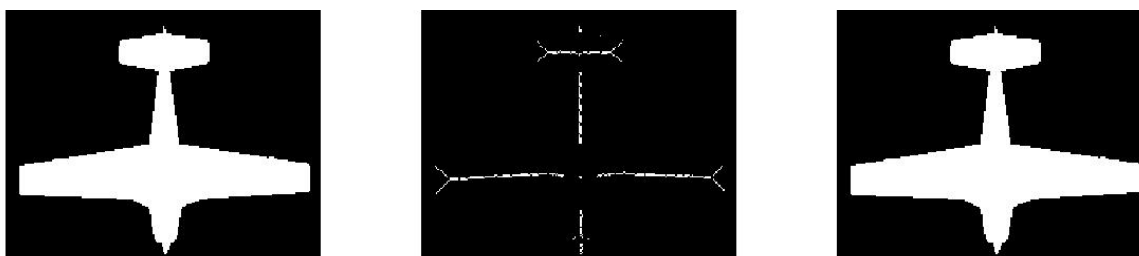


Figure 5

The next three tests were designed to check the algorithm's capability for handling multiple blobs in an image. We chose not to perform explicit image segmentation since the algorithm was designed to handle any number of disconnected blobs.

Multiple Blob Images

The first image in Figure 6 is an example of an image with a set of connected blobs. We were justified in not performing the segmentation of the image into its various components since our implementation of skeletonization computes the skeleton for the various blobs in the image itself. This test was run for the 8-connected distance metric.

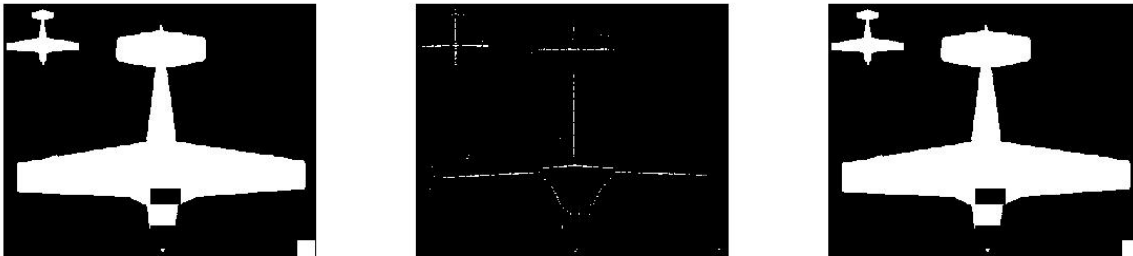


Figure 6

Images with blobs that touch at just one point

The first image in Figure 7 is an example of 2 blobs that just meet at one pixel. This reduces to a single connected blob. However we wanted to check if the algorithm computes the right maximum ball skeleton for such images. By looking at the skeleton in the second image of Figure, we see that the algorithm can correctly handle such cases. This test was run for the 8-connected distance metric.



Figure 7

Images with blobs separated by just one pixel

The first image in Figure 8 is an example of 2 blobs that are separated by just one pixel. By looking at the skeleton in the second image of Figure 8, we see that the algorithm can correctly handle such cases. This test was run for the Euclidean distance metric.

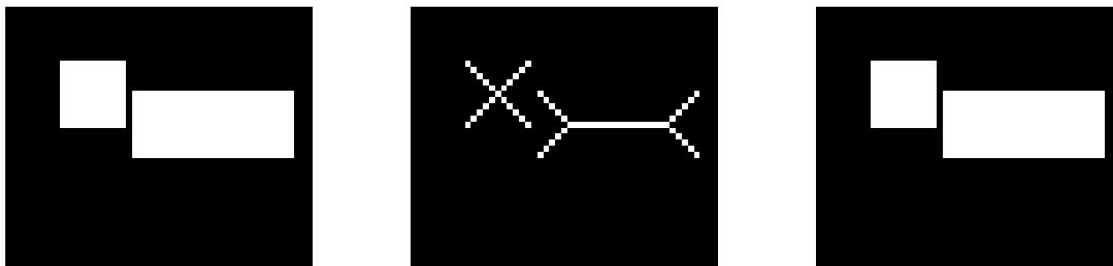


Figure 8

Skeleton Image

We wanted to check if the skeletonization of a skeleton resulted in the skeleton itself. Hence we ran this test. We used the skeleton (first image in Figure 9) produced from the test above as the input image. The results as seen in Figure 10 were as we expected, we cannot find any compression on an image which represents a skeleton. This test case was run for the 4-connected distance measure.



Figure 9

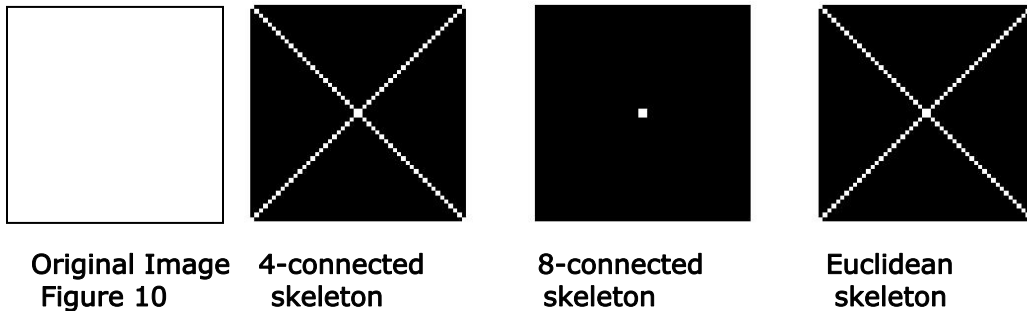
IV. ANALYSIS AND DISCUSSION

During the course of executing this project, we made the following observations.

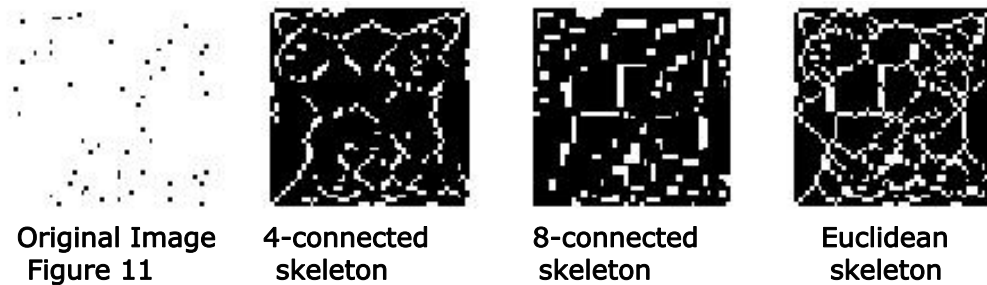
- Skeletonization is invertible
We can reconstruct the *exact* original image from the skeleton.
- Upper bound on the number of iterations
In our proposed approach (Section II), we are trying to construct maximum balls of radius r , at each iteration. Notice that we have defined the stopping criterion as not being able to construct a maximum ball of radius r for any image pixel for a particular r . We can prove that this will occur either before or at the iteration, when radius $r = n$; where $n = \min(\text{Number_of_rows_in_Image}/2, \text{Number_of_columns_in_Image}/2)$. We do so by considering the extreme case when all pixels in an image are foreground pixels. This results in the image consisting of 1 connected rectangular blob of size $\text{Number_of_rows_in_Image} * \text{Number_of_columns_in_Image}$. The ball of largest radius that can be inscribed in such a blob will be centered at the centre of the blob, and have a radius r equal to either the number of rows or columns in the image, whichever is less. Hence, this value forms an upper bound on the number of iterations the algorithm has to perform in constructing the maximum ball skeleton for such an image. For any other case, i.e. an image where not all pixels are foreground pixels, we may achieve the stopping criterion before we have completed n iterations. In any case, we will never exceed n .

- Maximum Ball Skeleton is highly sensitive to noise

We observed that the maximum ball skeleton achieves high rates (lower ratios) of compression if there is more patterned redundancy in the image it is trying to skeletonize. We tested this hypothesis by first creating a 50-by-50 test image of all foreground pixels as shown in the first image in Figure 10. On skeletonization (images 2, 3 and 4 in Figure 10), we observe that the skeletons for all three distance metrics (4-connected, 8-connected and Euclidean distance) are highly regular and compact since each skeleton pixel can account for many image pixels owing to the high degree of redundancy present.



We now add random 'pepper' noise to the original 50-by-50 image by generating 50 random locations between (1, 1) and (50, 50) and setting the pixel value to 0 at those locations as shown in image 1 of Figure 11. On computing the skeletons again, we find that the regular structure that the skeleton had is replaced with highly irregular skeleton pixel locations (images 2, 3 and 4 in Figure 11). This in turn decreases the compression rate. Hence, the introduction of even a small amount of noise destroys a lot of the redundancy and this change must be captured by the skeleton. The skeleton does this by adding more pixels.



- Affine Transform Invariance

We considered the behavior of the maximum ball skeleton under the affine transforms – translation, scaling and rotation. Translating the skeleton in the image plane does not affect the shape of the object reconstructed, i.e. the object is reconstructed perfectly but is translated by the exact amount by which the skeleton was translated. In the case of scaling, we found that if we resize the

skeleton by a factor s and scale the quench function values for each skeleton pixel by the same s , the reconstructed image shows no difference. This could be used as a method for finding an object in an image irrespective of its size, given its maximum ball skeleton at some size. We reconstruct the image for different scale values and match the resulting object with the objects in the image. When we arrive at a match, we know that the object is present at that scale in the image. The shape of the reconstructed image is not preserved under rotation of the skeleton. We feel this can be explained on the fact that, for image reconstruction, the maximum ball reconstructed by each skeleton pixel is not a perfect circle, i.e. it is not symmetric about all axes passing through its centre. Due to this asymmetry, when a rotated pixel is re-grown, it need not be in the same orientation it was in when that skeleton pixel captured it as part of its maximum ball.

V. CONCLUSION AND FUTURE WORK

We verified, as we implemented this project, that the maximum ball skeleton can be used as a lossless compression technique. Our observations also showed that it is possible to match objects at different scales using the same maximum ball skeleton. Hence, it has uses in generic object recognition. We would like to explore this area in further work.

VI. REFERENCES

- [1] '*A Transformation for Extracting New Descriptors of Shape*', Models for the perception of Speech and Visual form, W. Wathen-Dunn, Ed. Cambridge, MA: MIT Press, 1967, pp. 362-380
- [2] '*Skeleton-based Modeling Operations on Solids*'; Proceedings of the fourth ACM symposium on Solid modeling and applications, Pages: 141 - 154, 1997 Duane W. Storti, George M. Turkiyyah, Mark A. Ganter, Chek T. Lim, Derek M. Stal.
- [3] Hilditch CJ, '*Linear skeletons from Square Cupboards*'; In Machine Intelligence 4, Melzer b, Michie D (eds.).Edinburgh University Press, Edinburgh, 1969, pp 403-420.
- [4] www.cs.princeton.edu/courses/archive/fall03/cs597D/lectures/medial.pdf