Final Project

Content-Based Image Retrieval (CBIR) Using Barcode

Sujeev Uthayakumar
Faculty of Engineering
Ontario Tech University
100744194
sujeev.uthayakumar@ontariotechu.net

Zirak Mughal
Faculty of Engineering
Ontario Tech University
100749132
zirak.mughal@ontariotechu.net

Abdullah Waseem
Faculty of Engineering
Ontario Tech University
100748123
abdullah.waseem@ontariotechu.net

Jasarat Siddiqui
Faculty of Engineering
Ontario Tech University
100619789
jasarat.siddiqui@ontariotechu.net

***Abstract*— The process known as Radon transformation has been typically used to create barcodes which are used to tag medical images. This process works by taking a sampled image and transforming it into several projections (usually 4 or 8). These projections are then used to generate barcodes. [3] The generated barcode is then compared with the MNIST database of images using the Hamming distance between the input barcode and of each image on the MNIST database. Database images with a Hamming distance below a set threshold are outputted. Both of these processes were used in order to generate algorithms. In this paper, we propose two different algorithms, each with its own designated task. The first algorithm used to create a barcode for each 28-by-28-pixel image provided in the MNIST dataset. The second algorithm is used to then take that corresponding barcode and then utilize it to search for the most similar image in the given dataset. This is done by using the comparing the barcode of the query image with other barcodes in order to find the most similar image. Subsequently, experiments were conducted to report the retrieval accuracy of the algorithms, and the algorithm complexity was analyzed based on Big-O-Notation.**

I. Introduction

Generally, searching for an image in a set database is typically done through something known as Content-Based Imaged Retrieval (CBIR). Content-Based Image Retrieval works with the image itself rather than assigning specific text or data to it to search for it. Earlier CBIR systems identified a certain image by its description of visual characteristics like color, shape, texture, and other significant physical details. In 2015, an idea was proposed to use Radon barcode for image retrieval system in the medical field [3]. This was inspired by the many other products that use barcodes in some shape or form. Radon barcode is binary code that is generated by Radon Transformation and uses projection angles to tag an image. Using Radon barcodes is much more efficient than previous Content-Based Image Retrieval methods since it is a lot easier to search for specific images using Hamming distance. Hamming distance between two-bit strings that are equal length is the number of positions where corresponding bits are different, thus the images that are most like one another have the lowest Hamming distance [4].

II. Algorithms

A. MNIST Dataset Barcode Generation

The radon barcode generation algorithm works by creating projections for the angles 0, 45, 90 and 180 degrees for the chosen image. These projections are created and for this array a threshold mean is found. Where after, it is binarized to 1’s and 0’s where if the number were larger than the mean it would be set to 1 and 0 otherwise. After, the binarization of this array it becomes appended to the RBC array. After, the addition of this array the while loop will create a new angle through $\theta \leftarrow \theta + 180/n_\theta$. This will iterate through depending on the number of projections chosen, for this specific algorithm the while loop will run 4 times. Finally ending with a binary array with the dimension size of 112 x 1. This array can be reduced in size, with approximately seven digits on either size to retain accuracy. Ultimately, our final barcode remained at 112 x 1 due to the small dataset size. But for a larger dataset, the algorithm run time would decrease with a smaller binary array comparison.

Radon projections is the staple of the algorithm 1, where it focuses on finding the sum of an array in specific angles. For 4 projections, as previously state it will occur at 0, 45, 90 and 180. The figure below showcases, a simple implementation of radon projections for a 3 x 3 array. In our specific case, we will be working with a 28 x 28 down sampling size. Which ultimately means, that our array will be far larger. Our radon projections per angle will have a 28 x 1. With our full RBC array having an array size that can be denoted by:

$$RBC = Downsampling \ x \ Number \ of \ Projections$$

Which means that the RBC array will have a size of 112 x 1, after all four projections are appended.

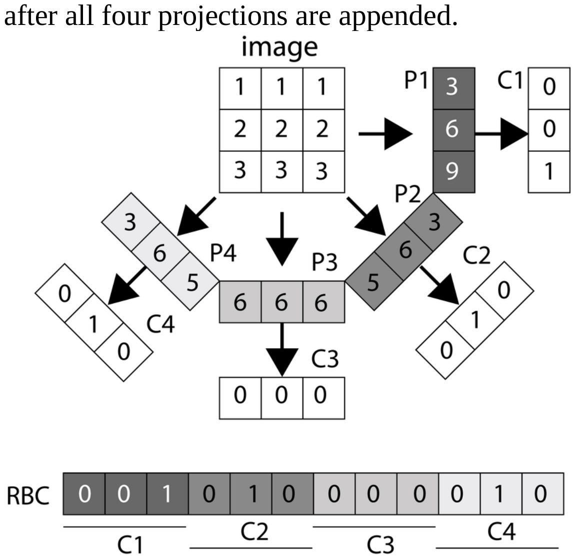


Figure 1 *Radon Barcode, Projections (P1, P2, P3, P4) binarized after radon projections are created. The binarization is dependent on the threshold mean.*

Algorithm 1 Radon Barcode Generation [9]

Import all libraries
Initialize Radon Barcode r $\leftarrow \theta$
Set down sampling size $R_N \leftarrow 28$
Set number of projection angles $n_\theta \leftarrow 4$
for i in range of 60,000 images:
    Initialize angle $\theta \leftarrow 0$
    Get query image I
    Downsample I: I = Normalize(I,R_N)
    while $\theta < 180$ **do**
        Get all projections **p** for θ
        Find typical value $T_{\text{typical}} \leftarrow \text{median}_i(\mathbf{p}_i)|_{p_i \neq 0}$
        Binarize projections: $\mathbf{b} \leftarrow \mathbf{p} > T_{typical}$
        Append the new row r $\leftarrow$ append(**r**, **b**)
        $\theta \leftarrow \theta + 180/n_\theta$
    end while
Create new data frame row
Send the full data frame to the excel file
Store RBC in Hamming array for comparison
Return **r**

```
[[0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
  1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.
  1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Figure 2 *Sample Barcode Generation*

Figure 1 showcases a potential radon barcode of an image. As you can see, it is a binary array of size 112 x 1, making it a perfect candidate for the Hamming distance formula.

B. Radon Barcode Comparison Using Hamming Distance

Similarity and error detection are generally found by calculating the frequency of dissimilarity between two data sets. These principles are applied when calculating Hamming Distance. It being a metric for comparing two binary datasets of equal length, Hamming Distance is the number of positions in which the two bits are different [4].

In our application Hamming Distance is fundamentally calculated by comparing the least significant bit in each RBC, if the values differ; a value is added to the counter. Then the least significant bit is popped from each RBC and a Boolean false is appended to each RBC, shifting them to the right. These orders of operations are performed until a Boolean false value is present in each index of each RBC. Shown in *figure 3* is the Hamming Distance of two eight-character arrays.

A_1 = [0,0,1,0,0,1,0,1], A_2 = [1,0,0,1,0,0,0,1]

00100101
10010001

10101010 counter = 0

00010010
01001000

01011010 counter = 0

00001001
00100100
___
00101101 counter = 1

00000100
00010010
___
00010100 counter = 1

00000010
00001001
___
00001011 counter = 2

00000001
00000100
___
00000101 counter = 3

00000000
00000010
___
00000010 counter = 3

00000000
00000001
___
00000001 counter = 4

***Figure* 3** *Hamming distance calculation of two binary arrays*

Hamming Distance provides a final numerical value to the amount of dissimilarity between a set of 2 RBC but to determine the amount of dissimilarity to allow is solved through setting a threshold value [4]. Each Hamming Distance is compared to a set threshold value, if the Hamming is below the threshold value, then the input, and comparison RBC represent similar images [7].

Image storage and retrieval efficiently is address as images are retrieved from a dataset of Radon Transformed images and appended to a dictionary. Appending the comparison images to a dictionary allows for image retrieval based on index rather than iteration over previously revised images proceeding an images pair presenting a Hamming distance below the threshold [4].

Algorithm 2 Hamming Distance Comparison [7]

1: Import all libraries
2: Initialize variables to find minimum Hamming distance
3: Initialize average variable
4: **for** i in the range of 100 images:
5: Initialize random value x
6: **for** j in range of 100 images:
7: Calculate hamming between random x value, and value j
8: **if** calculated hamming distance is less than potential store new hamming distance
9: **if** the x label matches with the potential j value add one to the average
10: **return** average
11: **print** average hit ratio

```
The testing label is:  1  and the image number is:  25003
The potential label is:  5  and the image number is:  0  and the Hamming value is  0.25
The potential label is:  0  and the image number is:  1  and the Hamming value is  0.16071428571428573
The potential label is:  1  and the image number is:  3  and the Hamming value is  0.08928571428571429
The potential label is:  0  and the image number is:  34  and the Hamming value is  0.07142857142857142
The potential label is:  1  and the image number is:  67  and the Hamming value is  0.0625
The potential label is:  1  and the image number is:  77  and the Hamming value is  0.05357142857142857
The potential label is:  1  and the image number is:  78  and the Hamming value is  0.026785714285714284
The potential label is:  1  and the image number is:  466  and the Hamming value is  0.008928571428571428
```

Figure 4 *Sample Hamming Distance Comparison, where algorithm takes the minimum*

In *figure 4* Hamming distance comparison algorithm is applied with an iteration of 100 images being compared via Hamming Distance to return an average hit ratio, the measurement of accuracy that the RBC generation and Hamming Distance comparison provide.

III. Comparison & Analysis

Retrieval accuracy of our searching algorithm in terms of hit ratio varies but in most cases came around 80%. While the accuracy can be increased with more iterations of the algorithm which would be the loop being run more times over but that would also increase run time. Retrieval accuracy can be increased in terms of hit ratio but that would lead to longer run-times.

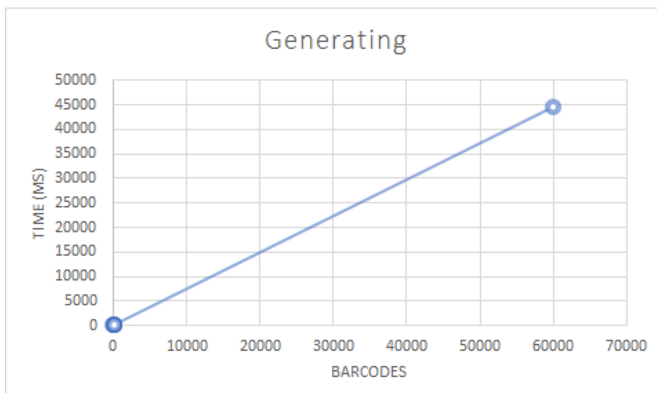


Figure 5 *Barcode Generation Time Comparison Graph*

Algorithm 1 Radon Barcode Generation: Big O Analysis

1: Import all libraries
2: Initialize Radon Barcode r ← θ
3: Set down sampling size R_N ← 28 } 1
4: Set number of projection angles n_θ ← 4
5 **for** i in range of 60,000 images:} ⟶ **(n-1) – 0 +2⟶O(n)**
6: Initialize angle θ ← 0
7: Get query image *I* } 1
8: Downsample I: I = Normalize(I,R_N)

9: **while** θ < 180 **do** } ⟶ **log(n)** ⟶ **O(log(n))**
10: Get all projections **p** for θ } ⟶ 1
11: Find typical value $T_{\text{typical}} \leftarrow \text{median}_i(\mathbf{p}_i)|_{p_i \neq 0}$
12: Binarize projections: $\mathbf{b} \leftarrow \mathbf{p} > T_{typical}$
13: Append the new row r ← append(**r**, **b**)
14: $\theta \leftarrow \theta + 180/n_\theta$
15: **end while**
16: Create new data frame row
17: Send the full data frame to the excel file
18: Store RBC in Hamming array for comparison
19: Return **r**
1

Big O Analysis= O(n*log(n))

The for loop is going to run in the range set in the algorithm which would in worst case scenario be 60000 so Big O is going to be n. The nested while loop will be log(n) since its nested within the for loop, it is going to be multiplied n*log(n) for Big O, the rest of the terms are constants so in the calculations are dropped. Figure 5 show roughly a linear relationship but since the range is so small it is hard to see the relation other than linear but once the n range is increased the algorithm will follow a n*log(n) relation.

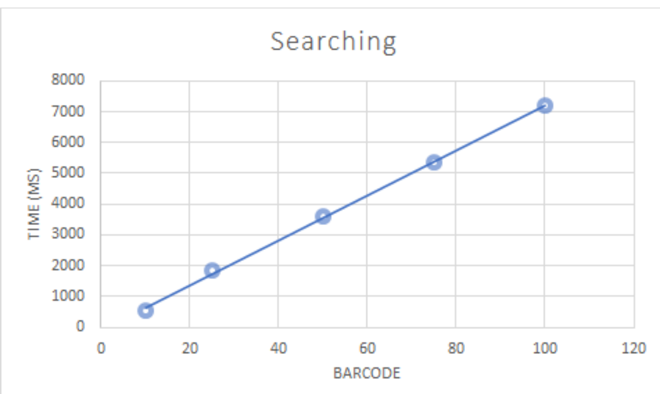


Figure 6 *Searching Algorithm Time Comparison Graph*

Algorithm 2 Hamming Distance Comparison: Big O Analysis

1: Import all libraries
2: Initialize variables to find minimum Hamming distance
3: Initialize average variable
1
4: **for** i in the range of 100 images:**}⟶ (n–1)-0+2 ⟶O(n)**
5: Initialize random value x } ⟶ 1
6: **for** j in range of 100 images: **}⟶ (n–1)-0+2 ⟶O(n)**
7: Calculate hamming between random x value, and value j
8: **if** calculated hamming distance is less than potential store new hamming distance
9: **if** the x label matches with the potential j value add one to the average
10: **return** average
1
11: **print** average hit ratio

Big O Analysis: O(n^2)

The for loop is going to run in the range set in the algorithm which would in worst case scenario be 100 so Big O is going to be n. The nested for loop will also run for n times since its nested within the for loop, it is going to be multiplied n*n for Big O, the rest of the terms are constants so in the calculations are dropped. Figure 6 show roughly a linear relationship but since the range is so small it is hard to see the relation other than linear but once the n range is increased the algorithm will follow a O(n^2) relation.

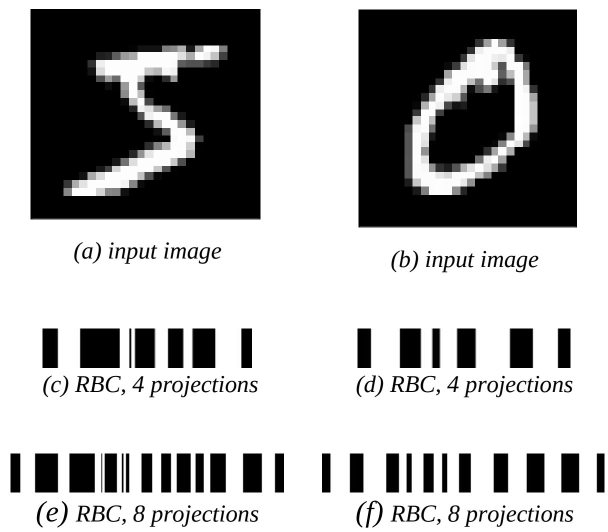


Figure 7 *Radon Barcodes with 4/8 Projection Angles*

IV. Results Comparison

In this section we cover and compare the time elapsed of both the Radon Barcode generation algorithm and the Hamming distance comparison search and retrieval. 5 types of image query tests, each involving a varying amount query barcode were performed.

A. Radon Barcode

In the case of the Radon Barcode generation the tests were benchmarked through the average time elapsed in 3 tests to generate 10, 50, 75, 100 and 60,000 4x28 projection Radon Barcodes representing the respective number of images in the MNIST database.

Table 1. *Radon Barcode generation times*

Number of Radon Barcodes Generated	Time for Generation
10	0:00:05
50	0:00:12
75	0:00:15
100	0:00:19

60,000	0:44:68

Time for generation increases with number of radon barcodes as seen in *Table 1.* This reflects the algorithms analyzed time complexity of O(n*log(n)) with relatively low time for generation with a low number of barcodes generated compared to a higher time for generation as the number of barcodes reaches a larger number.

B. Radon Barcode Retrieval and Hamming Distance Comparison

In the case of the searching algorithm, tests performed were to benchmark the average time elapsed to iteratively compare the set number of radon barcodes to a dataset of 60,000 MINIST image radon barcode. As well as the accuracy of comparison, represented through a hit ratio percentage.

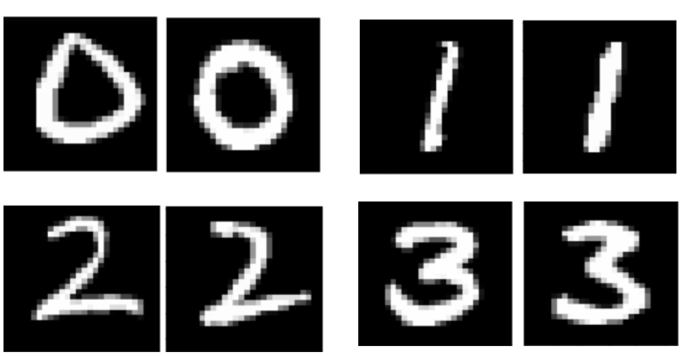
Table 2. *CBIR search result times*

Number of Radon Barcodes Compared	Time for Comparison	Hit Ratio
10	0:00:55	93%
25	0:01:84	84%
50	0:03:60	78%
75	0:05:38	84%
100	0:07:20	77%

In *Table 2* the time complexity of the searching algorithm is shown in that due to the lower number of barcodes being compared that the time for comparison is low. Reflecting on the hit ratio, it presents a very high accuracy, this is due to a low set Hamming Distance threshold value which ultimately only allows high similarity Radon Barcodes to pass and thus high similarity images.

V. Search Results

In this section input images will be compared to their output counterparts by class of image, representing the 9 classes present within the MNIST databases.



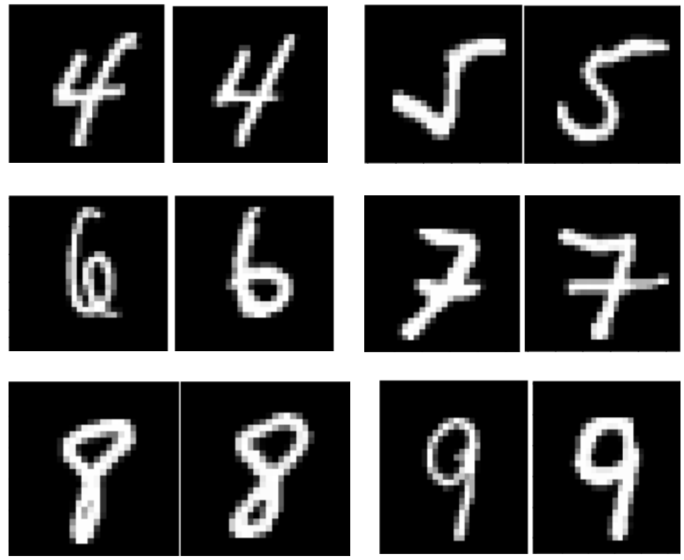


Figure 8. *Input and output image pair for each image class, input: left, output: right*

In *Figure 8*, on the left are presented input query images from each class of the of the MINIST database and on the right of each image pair is the is the image associated with the Radon Transformed barcode of the lowest hamming distance and most similarity for the specified input image.

VI. Conclusions

Therefore, the process of retrieving one image from a specific archive of images is a useful and demanded task. Time and time again the use of radon transformation is proved useful, particularly in the medical industry where it is used to create barcodes that tag medical images [10].

Ultimately, with an average hit ratio of 77% for 100 barcodes the radon projection-based barcode generation and success can be deemed a success. The highest average hit ratio reached 83% on three test trials. For improved performance, increasing the number of projections is a potential approach that can be taken. The sacrifice that is made with increasing the number of projections is increasing the run time. Tests showed that when projections are increased, the accuracy can increase by at least 6%. Although, the accuracy increases the array size increases meaning that generation and searching run time will also increase.

In this work, two algorithms were generated, each with its own designated task. One algorithm to create a barcode for each 28x28 pixel image in the MNIST dataset. The second algorithm to then take that barcode and use it to search for the most similar image in the given dataset. The process worked by comparing the barcode of the query image with the other barcodes to find the most similar image. Afterwards, experiments were conducted to calculate the retrieval accuracy of the algorithms. The algorithm complexity was also analyzed based on Big-O-Notation.

VII. References

[1] - E. W. Wesstein, "Radon Transform," from Wolfram MathWorld. [Online]. Available: https://mathworld.wolfram.com/RadonTransform.htm. 27-Mar-2021.[1] - E. W. Wesstein, "Radon Transform," from Wolfram MathWorld. [Online]. Available: https://mathworld.wolfram.com/RadonTransform.htm. 27-Mar-2021.

[2] -"Evolutionary projection selection for Radon barcodes," *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7743771&isnumber=7743769. 26-Mar-2021.

[3] - H.R. Tizhoosh, "Barcode annotations for medical image retrieval: A preliminary investigation," in IEEE ICIP, 2015. 18-Mar-2021

[4] - N. Raut, What is Hamming Distance?, 31-Dec-2018. [Online]. Available: https://www.tutorialspoint.com/what-is-hamming-distance#:~:text=Hamming%20distance%20is%20a%20metric,d(a%2Cb). 20-Mar-2021.

[5] - Pypi, "Radon," *PyPI*, 09-Mar-2021. [Online]. Available: https://pypi.org/project/radon/. 31-March-2021.

[6] - "Radon Barcodes," Kimia Lab. [Online]. Available: https://kimialab.uwaterloo.ca/kimia/index.php/radon-barcodes/. 01-Apr-2021.

[7] - "Scipy Spatial.Distance Hamming,"- SciPy v1.6.2 Reference Guide. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.hamming.html. 28-March-2021.

[8] S.K. Shandilya and N. Singhai, "A survey on content based image retrieval systems," International Journal of Computer Applications, vol. 4, 2010. 22-Mar-2021

[9] - S. Rayhnamayan and H. R. Tizhoosh, "EVOLUTIONARY PROJECTION SELECTION FOR RADON BARCODES," in IEEE CEC 2016. 18-Mar-2021

[10] - X. Zhang and S. Zhang, "Hashing-based large-scale medical image retrieval for computer-aided diagnosis," Machine Learning and Medical Imaging, 19-Aug-2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128040768000086. 20-Mar-2021.