SRH University Heidelberg

Task 2:
To estimate the similarity between two input images by calculating the cosine similarity between their histograms of oriented gradients.

Author:	Nikhil Kumar
Matriculation Number:	11037894
Email Address:	nikhil.kumar@stud.hochschule-heidelberg.de
Date of Submission:	26th February 2024

Under the Guidance of Professor
Dr. Ing Milan Gnjatovic

Aim of the Task:

To compute a histogram of oriented gradients for a given image, Where input parameters are: grayscale image, number of chain code directions, and grid dimension.
Then, it estimates the similarity between two input images by calculating the cosine similarity between their histograms of oriented gradients.

Objective:

The objective of the cosine similarity function is to provide a reliable and effective method for quantifying the similarity between two vectors, which can be used in a wide range of applications.

Introduction :

Cosine similarity is a metric used to determine how similar two images or vectors are to each other in a high-dimensional space. It measures the cosine of the angle between two vectors, providing a measure of orientation rather than magnitude. In other words, it calculates the cosine of the angle between two vectors projected in a multi-dimensional space. Cosine similarity is widely used in information retrieval, natural language processing, and recommendation systems to quantify the similarity between documents, texts, or items. It is particularly useful when dealing with sparse datasets or when the magnitude of the vectors is not relevant to the similarity measure.[1a]

Working and Procedure steps to find the Cosine Similarity of two images:

SOBEL FILTER:

Initially, we must determine both the magnitude of the gradient vector and the orientation angle (Theta) to achieve this, we employ the Sobel Filter.

The Sobel Filter serves the purpose of edge detection, operating by computing the gradient of the image intensity for each pixel. It utilizes a specific formula to compute both the vertical edges (Sx) and horizontal edges (Sy).

Vertical Edge:	Horizontal Edge:
Sx={{-1, 0, 1},	Sy = {{-1,-2,-1}
{-2, 0, 2},	{0, 0, 0}
{-1, 0, 1}};	{1, 2, 1}}

Utilizing the aforementioned formula for both vertical and horizontal edges on image pixels yields the magnitude of the gradient vector, denoted as Gx for horizontal edges and Gy for vertical edges.

The obtained gradient vector can be represented as the ordered pair:
g(x,y) = (||g(x,y)||,theta(x,y))

(||g(x,y)||, is the magnitude of the gradient vector, which is approximated as
||g(x,y)|| = sqrt(gx * gx + gy * gy);

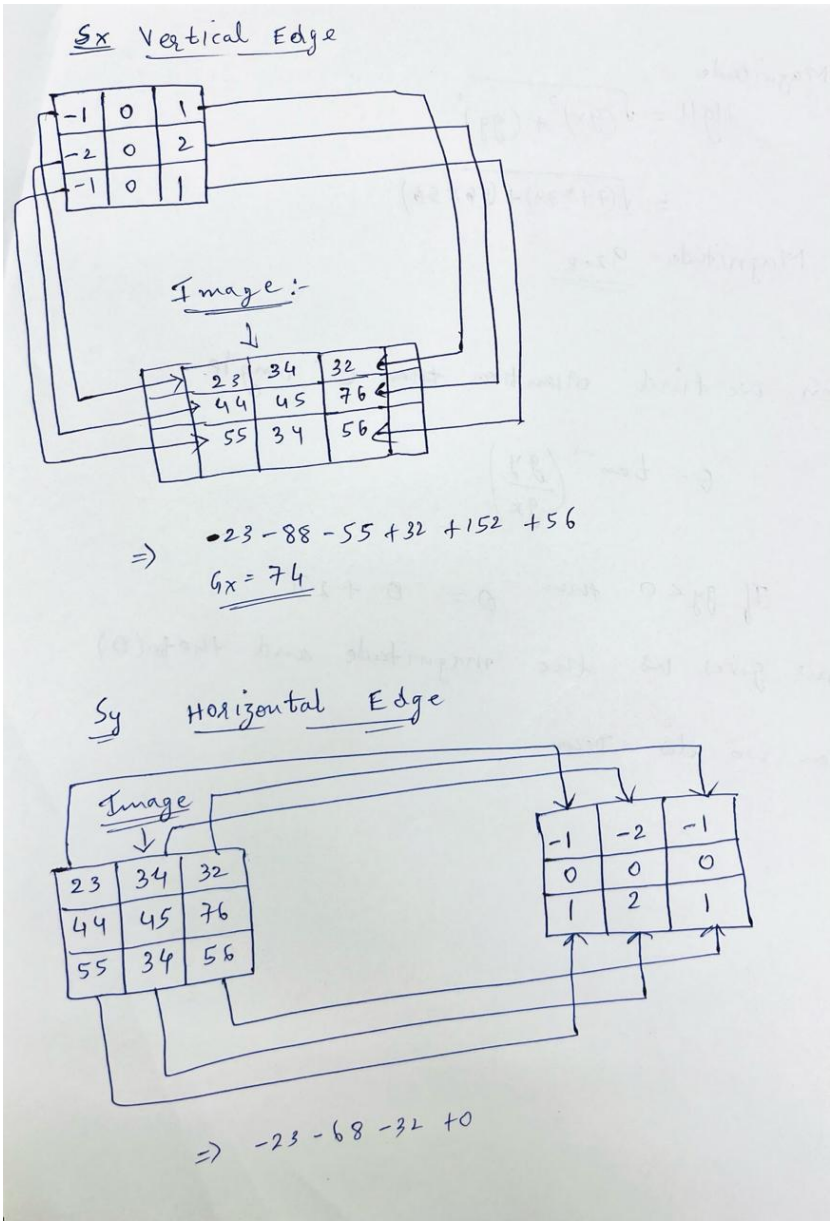
Then we need to find the orientation (theta), which is found out by :

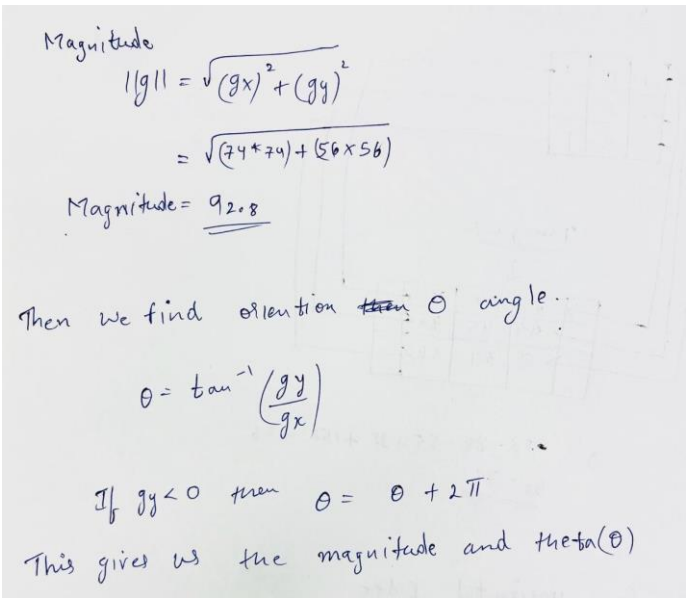
$$\theta(x,y) = \begin{cases} \arctan\dfrac{g_y(x,y)}{g_x(x,y)}, & \text{if } g_x(x,y) > 0\,, \\ \arctan\dfrac{g_y(x,y)}{g_x(x,y)} + \pi, & \text{if } g_x(x,y) < 0\,, \\ \frac{\pi}{2} \cdot sgn(g_y(x,y)), & \text{if } g_x(x,y) = 0\,. \end{cases}$$

Next, we utilize the chain code directions (n) to determine the index, known as the decomposition of the gradient vector.

Index = (theta * n) / (2 * PI)

Here's an illustration of a handwritten image demonstrating the application of the Sobel filter to every pixel of the image and the subsequent computation of magnitude. This process enables us to derive histograms of oriented gradients.





Normalizing The histograms of oriented gradients:

Normalizing of HOG (Histogram of Oriented Gradients) refers to a process where the computed HOG descriptors are scaled to make them less sensitive to variations in illumination and contrast. This normalization step helps in improving the robustness of the HOG feature representation for object detection and recognition tasks.

Normalization of HOG helps in making the feature representation more invariant to changes in lighting conditions and enhances the discriminative power of the features, thereby improving the performance of object detection and recognition algorithms.

Cosine Similarity:

Cosine similarity works by comparing the angle between two vectors in a high-dimensional space. The angle between two vectors is calculated based on the dot product of the two vectors and their magnitudes.
Let's consider two vectors A and B in a 2-dimensional space as an example. The angle between these two vectors can be calculated using the dot product formula:
$A . B = ||A|| ||B|| \cos(\text{theta})$. [1b]
where A. B is the dot product of vectors A and B, ||A|| and ||B|| are the magnitudes of vectors A and B, respectively, and theta is the angle between the two vectors.
Rearranging the above equation, we get:
$\cos(\text{theta}) = (A . B) / (||A|| ||B||)$
The cosine similarity between two vectors A and B can be calculated by taking the cosine of the angle between them.
$\text{cosine similarity}(A, B) = \cos(\text{theta}) = (A . B) / (||A|| ||B||)$

The value of cosine similarity ranges from -1 to 1. If the two vectors are identical, the cosine similarity is 1. If the two vectors are orthogonal (i.e., have no similarity), the cosine similarity is 0. If the two vectors are opposite, the cosine similarity is -1. [1c]

Explaining the Program Functionality(What is going in the code):

As explained in the work, each step is defined as a function in the program and it is explained below. (Program File: Project_SubTask2.pde)

Grey Scale Function:

The grayscale function takes a single parameter, the input image. Within this function, an image named "img_G" (a placeholder or dummy image) is created using the createImage data type, and pixels are loaded for both the original image and the newly created image. Subsequently, a for loop is employed to iterate through the columns (y) and rows (x) of the image. The index is calculated for each pixel using the formula x + y * image width. RGB components of the image pixels are then extracted by their respective indexes using the predefined keywords red, green, and blue.

Next, the color image is converted to a grayscale image using the luminosity method, which employs the formula: 0.21 * red channel pixels + 0.72 * green channel pixels + 0.07 * blue channel pixels. This formula preserves the luminance information of the original image while discarding color information, resulting in a grayscale representation. Finally, the pixels of the "img_G" image are updated with the calculated grayscale values, and the grayscale image is returned.

Sobel Filter:

1. float[][] filter1 and float[][] filter2 are 2D arrays representing two Sobel filters, pivotal tools in image processing for edge detection. These filters, structured as small matrices, are convolved with the image to discern gradients, aiding in the identification of edges and contours.

2. The nested loops, encompassing (int y = 1; y < img.height - 1; y++) and (int x = 1; x < img.width - 1; x++), meticulously traverse each pixel within the image, excluding border pixels. This meticulous approach is essential as the Sobel filter necessitates a pixel neighborhood surrounding the focal pixel for accurate gradient calculation.

3. Within these nested loops, two pivotal variables, f1 and f2, are initialized to zero. These variables serve as repositories for the gradient components, diligently computed utilizing the Sobel filters.

4. Further nested loops (int ky = -1; ky <= 1; ky++) and (int kx = -1; kx <= 1; kx++), meticulously iterate over a 3x3 pixel neighbourhood cantered around the focal pixel. This meticulous traversal ensures the Sobel filters' application to each pixel's immediate vicinity.

5. Within these nested iterations, the variable index is meticulously calculated to facilitate pixel value retrieval from the image. This calculation meticulously determines the index of the pixel within the image's pixel array, factoring in the current pixel's coordinates and the offsets provided by ky and kx.

6. Pixel brightness at position (x + kx, y + ky) is meticulously acquired utilizing brightness(img.pixels[index]). This function meticulously returns the brightness value of the pixel, reflective of its luminance or intensity.

7. Subsequently, the computed gradient components f1 and f2 are meticulously updated. This involves meticulously multiplying the pixel's brightness with the corresponding values from the Sobel filters (filter1 and filter2), culminating in the accumulation of results. The indexing (ky+1, kx+1) ensures precise mapping of filter elements to the correct position within the pixel neighborhood.

8. Upon meticulous processing of all pixels within the neighborhood, the values of f1 and f2 meticulously represent the gradient components along the x and y directions at the focal pixel, crucial for subsequent edge detection and image analysis.

Gradient Vector:

1. **Calculation of Gradient Magnitude (mag):**

   The gradient magnitude, denoted as 'mag', is determined using the formula sqrt(f1 * f1 + f2 * f2). This calculation yields the Euclidean distance, or length, of the gradient vector, portraying the intensity of the edge present at the current pixel.

2. **Determination of Gradient Direction (theta):**

   The gradient direction, referred to as 'theta', is derived utilizing the atan2(f2, f1) function. This function computes the angle, measured in radians, between the positive x-axis and the gradient vector. It takes into account both the y-component (f2) and the x-component (f1) of the gradient vector. In instances where f2 is negative, indicating a downward-pointing gradient vector, 2π (representing a full circle) is added to the angle 'theta'. This adjustment ensures that the angle accurately reflects the direction of the gradient vector within the range of $[0, 2\pi]$.

3. **Mapping Angle to Histogram Bin (index array):**

   Next, the angle 'theta' is mapped to a corresponding bin within the histogram. Since histogram bins are utilized to represent gradient directions, it is necessary to discretize the continuous range of angles into a finite number of bins. The variable 'index array' is computed using the formula int(theta * n / (TWO_PI)), where 'n' signifies the number of bins in the histogram. This formula facilitates the mapping of the angle 'theta' to an integer index, effectively representing the histogram bin where the gradient magnitude will be aggregated.

Histogram Accumulation:

Finally, the magnitude (mag) of the gradient is added to the appropriate bin of the histogram. The variables x-axis and y-axis determine the position of the current pixel within the grid of cells used for histogram calculation.

The expression **main_array[y_axis * gridSize + x_axis][indexarray] += mag** accumulates the gradient magnitude (mag) into the histogram bin corresponding to the current pixel's position and gradient direction. In essence, this code snippet calculates the gradient magnitude and direction for each pixel in the image and adds the magnitude to the respective histogram bin based on the gradient direction

Cosine Similarity:

Cosine Similarity calculates the cosine similarity between two sets of vectors represented by 2D arrays (main_array1 and main_array2).

1. **Initialization:**

Three float variables are initialized: dot product to accumulate the dot product of corresponding elements in the two vectors, and mag1 and mag2 to accumulate the squared magnitudes of each vector.

2. **Compute dot product and magnitudes:**

Two nested loops iterate through the elements of main_array1 (assuming both arrays have the same dimensions).
In each iteration, the dot product is computed by multiplying the corresponding elements of main_array1 and main_array2 and adding the result to dot product.
The squared magnitudes of each vector are calculated by summing the squares of their elements and accumulating them in mag1 and mag2 respectively.

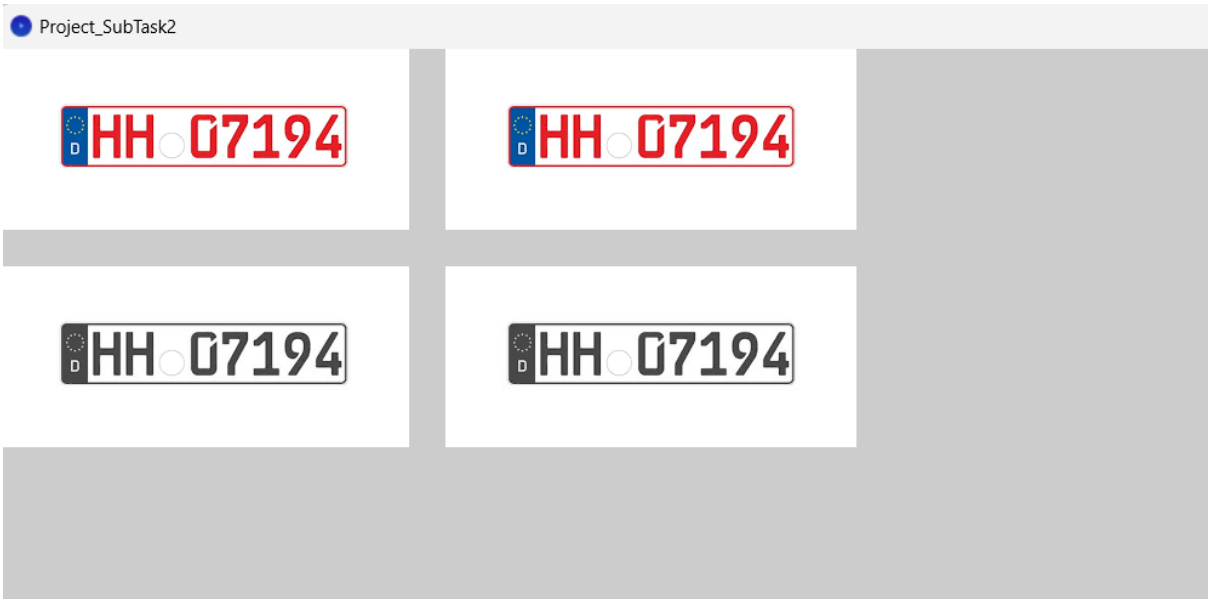
3. **Compute cosine similarity:**

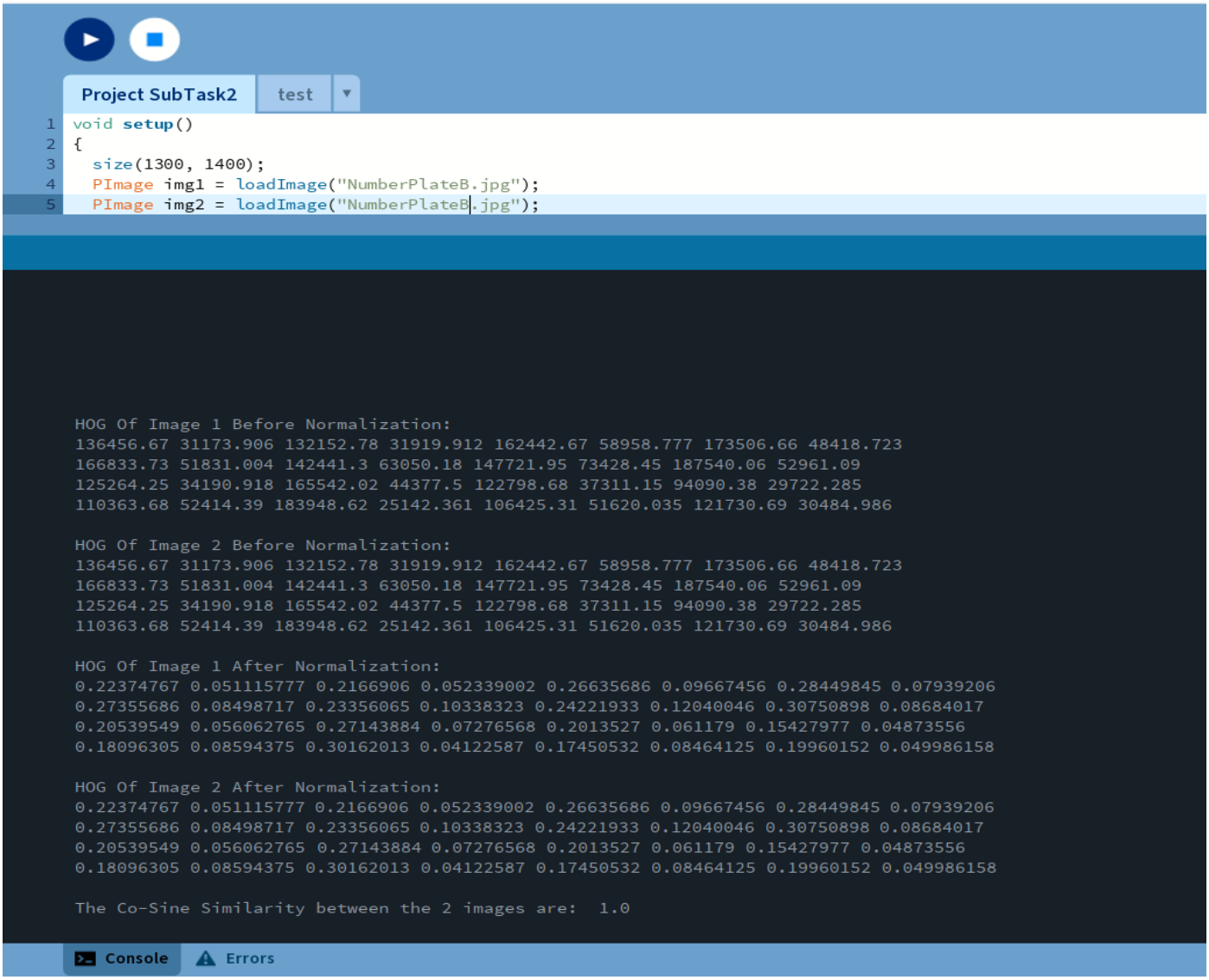
After computing the dot product and magnitudes, the cosine similarity is calculated using the **formula: dot product / (sqrt(mag1) * sqrt(mag2)).**

This formula computes the cosine of the angle between the two vectors, which is a measure of their similarity. If both vectors are identical, the cosine similarity will be 1.
If they are not identical, the cosine similarity will be 0.
If they are opposite, the cosine similarity will be -1.

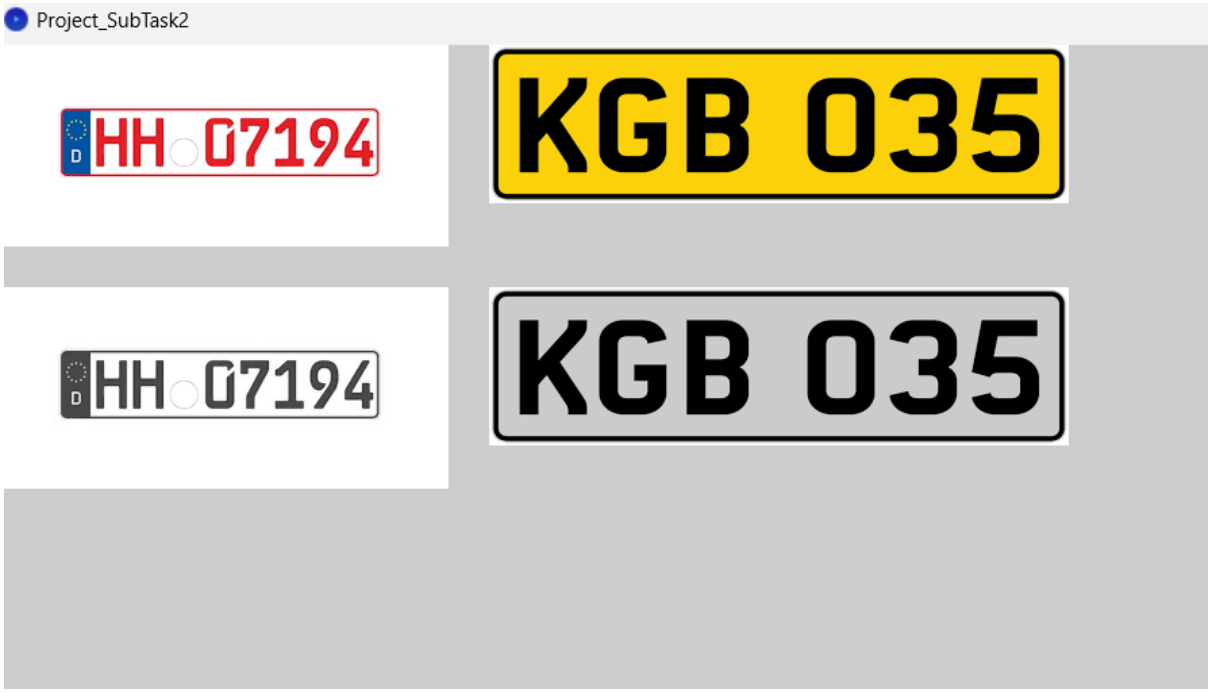
To handle cases where one or both of the magnitudes are zero (which would result in division by zero), a check is performed. If either magnitude is zero, indicating one of the vectors has no magnitude (i.e., it's a zero vector), the function returns 0, indicating no similarity.

Output Result 1: When Both the input images are the same



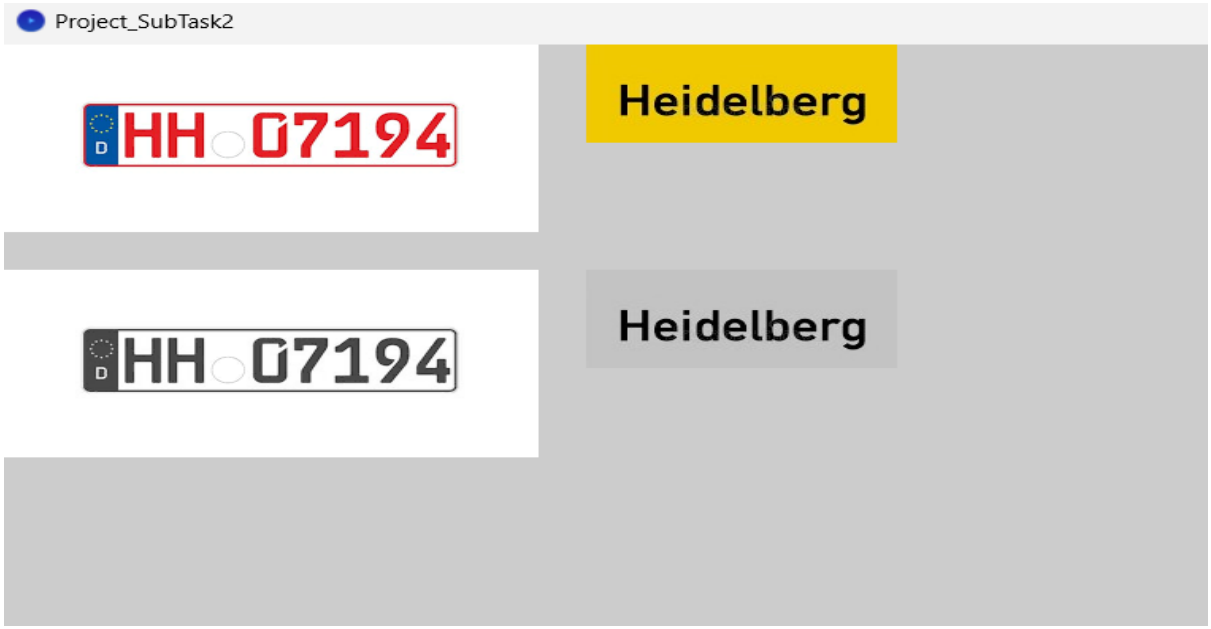


Output Result 2: When there are 2 different images.




```
HOG Of Image 1 Before Normalization:
136456.67 31173.906 132152.78 31919.912 162442.67 58958.777 173506.66 48418.723
166833.73 51831.004 142441.3 63050.18 147721.95 73428.45 187540.06 52961.09
125264.25 34190.918 165542.02 44377.5 122798.68 37311.15 94090.38 29722.285
110363.68 52414.39 183948.62 25142.361 106425.31 51620.035 121730.69 30484.986

HOG Of Image 2 Before Normalization:
268929.2 55812.586 374542.75 20903.535 337413.78 71336.81 605169.06 58593.258
229195.44 41823.38 436966.0 31189.4 244300.28 66337.52 632883.1 68526.22
259275.75 64243.523 584163.75 123024.59 261349.58 24174.357 376740.16 102121.56
250215.5 114911.484 594922.94 75935.914 192487.97 83491.43 398691.47 54349.38

HOG Of Image 1 After Normalization:
0.22374767 0.051115777 0.2166906 0.052339002 0.26635686 0.09667456 0.28449845 0.07939206
0.27355686 0.08498717 0.23356065 0.10338323 0.24221933 0.12040046 0.30750898 0.08684017
0.20539549 0.056062765 0.27143884 0.07276568 0.2013527 0.061179 0.15427977 0.04873556
0.18096305 0.08594375 0.30162013 0.04122587 0.17450532 0.08464125 0.19960152 0.049986158

HOG Of Image 2 After Normalization:
0.16334188 0.033899378 0.22748931 0.012696363 0.20493796 0.043328464 0.36756682 0.0355883
0.13920844 0.025402632 0.26540387 0.01894378 0.1483828 0.040292002 0.38439977 0.04162137
0.15747859 0.03902015 0.35480866 0.07472252 0.15873818 0.014682991 0.22882397 0.062026467
0.15197557 0.0697948 0.36134356 0.04612186 0.1169131 0.05071092 0.24215673 0.033010658

The Co-Sine Similarity between the 2 images are:  0.9477473
```

Output Result 3: When there are 2 different images.




```
HOG Of Image 1 Before Normalization:
136456.67 31173.906 132152.78 31919.912 162442.67 58958.777 173506.66 48418.723
166833.73 51831.004 142441.3 63050.18 147721.95 73428.45 187540.06 52961.09
125264.25 34190.918 165542.02 44377.5 122798.68 37311.15 94090.38 29722.285
110363.68 52414.39 183948.62 25142.361 106425.31 51620.035 121730.69 30484.986

HOG Of Image 2 Before Normalization:
22696.63 2449.494 3976.575 18610.3 25270.232 32852.65 29995.781 21096.094
7143.8574 911.77155 582.3144 10975.179 14725.291 35368.598 31390.34 19201.969
75210.76 65084.938 67629.22 77320.51 83365.96 37612.23 39084.336 69593.805
86747.055 68826.75 61417.824 78299.82 80584.336 32457.916 37711.797 89065.8

HOG Of Image 1 After Normalization:
0.22374767 0.051115777 0.2166906 0.052339002 0.26635686 0.09667456 0.28449845 0.07939206
0.27355686 0.08498717 0.23356065 0.10338323 0.24221933 0.12040046 0.30750898 0.08684017
0.20539549 0.056062765 0.27143884 0.07276568 0.2013527 0.061179 0.15427977 0.04873556
0.18096305 0.08594375 0.30162013 0.04122587 0.17450532 0.08464125 0.19960152 0.049986158

HOG Of Image 2 After Normalization:
0.079655334 0.0085966615 0.013956054 0.06531409 0.08868756 0.115298554 0.10527219 0.07403815
0.025071843 0.0031999229 0.0020436712 0.03851812 0.051679388 0.124128446 0.11016649 0.06739059
0.26395717 0.22841991 0.23734924 0.27136147 0.2925784 0.13200262 0.13716908 0.2442441
0.30444455 0.24155205 0.21554995 0.27479842 0.28281608 0.113913216 0.13235207 0.31258234

The Co-Sine Similarity between the 2 images are:  0.67856306
```

References:

1. Text Reference :

a) https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf
b) https://www.numerade.com/ask/question/i-need-specific-solution-please-consider-the-equations-for-two-planes-2x-3y-2-x-zyz3-determine-the-intersection-of-the-two-planes-if-it-exists-4-al-describe-the-geometric-properties-the-two--91165/
c) https://medium.com/@igniobydigitate/a-beginners-guide-to-measuring-sentence-similarity-f3c78b9da0bc
d) Lecture Notes.

2. Source Code Reference :

a) http://gnjatovic.info/imageprocessing/ip.code.snippets.txt
b) Lecture Notes.

3. Image Reference:

- https://fuwong.com/wp-content/uploads/2017/08/German-license-plate-101-Europlates-wiki-11.jpg
- https://t3.ftcdn.net/jpg/02/30/40/42/360_F_230404295_XkJmajXRI1eBj1DY0qZRD89mbzoX2Q0P.jpg
- https://thumbs.dreamstime.com/z/typical-german-yellow-city-sign-heidelberg-view-193020935.jpg?ct=jpeg