**PROJECT TOPIC** : Redrawing Handwritten Linear Sketches

**TEAM MEMBER :**

* Ahmet YOLDAŞ – 19050211052
* Sena NAL – 19050211025
* Alper Mert EKİCİ – 20050211060

**CODE EXPLANATION**

We apply Canny edge detection and Hough transform for line detection to perfectly transform a given image with hand-drawn linear drawings into a computer-drawn image.

**Canny edge detection** is used to detect edges and is a very successful operator. After reducing the noise with Gaussian, it detects the density and direction at the edges with the Sobel operator. Then, unreal edges are eliminated with non-maximum suppression and the detection is completed by connecting the edges according to the threshold values.

metin, yazı tipi, ekran görüntüsü, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

This section obtains a binary image by applying Canny edge detection thanks to the edge function. But first, we convert the image into grayscale then binarize it to make it only black and white pixels. Then we take the inverse of it, so the black parts are the white parts and vice versa. This is done because the operations we implemented are done when the pixels have the value of 1 which is white in binary, then we continue with the Canny edge detection. The Canny parameter specifies the properties of the Canny edge pollutant. Values ​​in the range [0.04 0.1] are equivalence values ​​that control the precision of determining features. 4 indicates the temperature of the Gaussian filter during edge change.

**Hough Transform** is a frequently used method to determine the existence and location of lines or shapes. Lines are represented parametrically in polar coordinates in the Hough transform. Canny takes an image with edge detection applied, calculates “r”, “theta” and “rho” values ​​for each pixel in the image, for all lines that can pass through this pixel, and collects votes for these lines in an accumulator. Highly rated lines are considered as real lines that are thought to exist in the image.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this section, Hough parameters in the image are obtained by applying the Hough transform. Then, The peaks in the matrix obtained as a result of the Hough transformation are determined and the end points of the line segments are calculated. The houghlines function extracts lines of a certain spacing and length. The FillGap parameter is used to join different line segments. The MinLength parameter specifies the minimum line length that will be accepted. If we don’t apply MinLength parameter, then it may cause distorted result with many small lines. By doing this we prevented that outcome.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

This loop takes the endpoints of each line segment in the LinesIm structure (point1 and point2), assigns them to variables named ab and cd, then adds these points to the matrices Input1 and PointsGroup. While connecting two points to the Input1 matrix, two lines for each line segment, it adds the start and end points and the indices of the line segment corresponding to these points to the PointsGroup matrix.

metin, ekran görüntüsü, yazı tipi, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

This loop calculates the distances of each point in the Input1 matrix to other points. The difference variable is assigned the distance value calculated using the Euclidean distance formula between two points. The calculated distance values ​​are added to the matrix called DifferenceArray respectively. In the second loop, ColIndex is used to store the offset values ​​in each row. It then adds the pad matrix to the left of the DifferenceArray matrix.

metin, ekran görüntüsü, yazı tipi, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

We used the loop here to identify similar points by comparing the differences between points with a certain threshold value. This similarity value (we set it to 70) is called the threshold value and can be adjusted depending on the user's application. The SamePoint matrix will be a Boolean matrix denoting similar points: 1(true) denotes similar point; 0(false) represents the dissimilar point.

metin, ekran görüntüsü, yazı tipi, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

In this part, we scan the SamePoint matrix and for each element with a value of 1 (true), we add the relevant index information to the Input2 matrix. The index information of each similar point is DifferenceIndex\_1 and DifferenceIndex\_2 values.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

This loop provides a control to remove redundant elements and duplicates from the B\_Input2 matrix through the Temp variable. Both loop steps delete columns depending on whether Temp is odd or even and the state of the flag variable.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

This part scans the B\_Input2 matrix and checks each element. If an element satisfies certain conditions (these conditions are b < ceil(row1/2) && b ~= DifferenceIndex\_1 && b ~= 0), that element is added to the DelNum matrix. These conditions are generally determined for pairs of similar points. As a result of this process, the DelNum matrix is ​​a vector containing the index information to be deleted.

metin, ekran görüntüsü, yazı tipi, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

This loop extracts the relevant rows from the B\_Input2\_2 matrix using the indices from the DelNum matrix.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

For each pair of similar points in the B\_Input2\_2 matrix, it finds the corresponding point from the PointsGroup matrix and adds it to the B\_Input2\_3 matrix.

metin, ekran görüntüsü, yazı tipi, el yazısı içeren bir resim

Açıklama otomatik olarak oluşturuldu

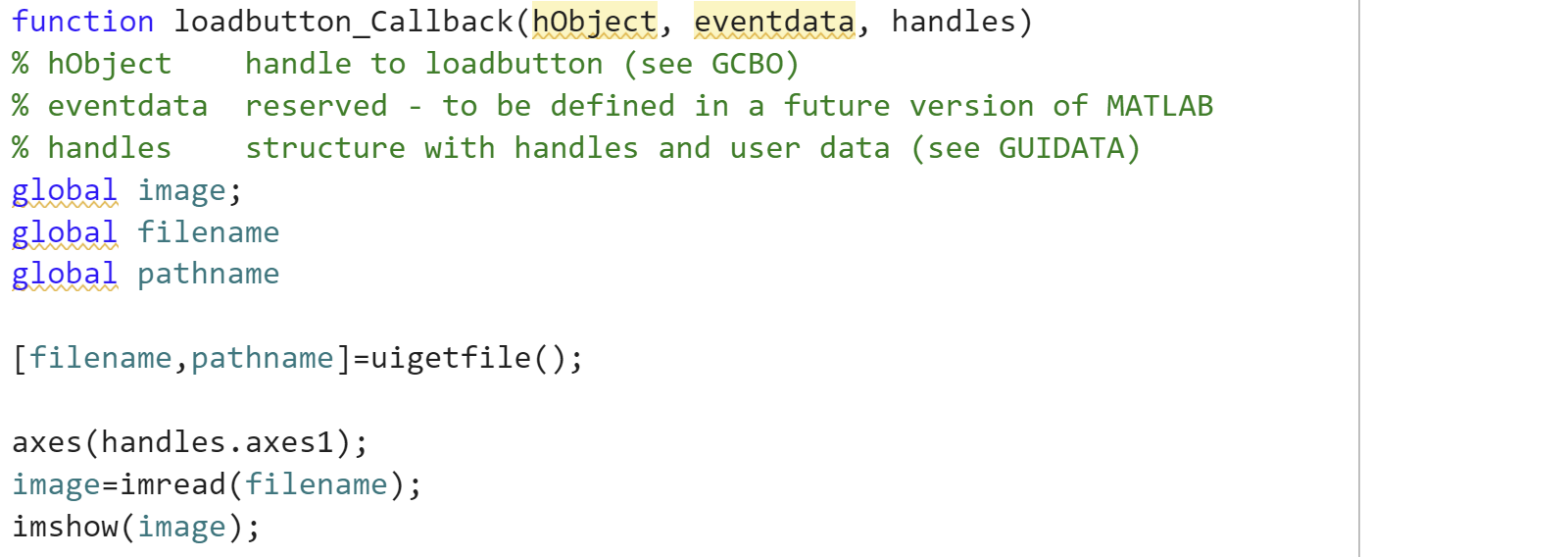
Here it loops through the recorded points in the B\_Input2\_3 matrix. By combining the coordinates of the regions corresponding to each pair of similar points (Output1\_x, Output1\_y and Output2\_x, Output2\_y), blue lines (using line coordinates) are drawn from each other.

**GUI**

While testing this application, we designed an interface in MATLAB where we can load images, see and save the results. Our interface consists of load, run, save exit buttons and the edit where we will write to save the image.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu



In the callback section of our load button, we defined image filename and pathname as global variables so that we can use them in other callbacks. We took the filename and pathname information of the file we uploaded with uigetfile() and displayed our original image in axes1.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

In the save and edit1 callback functions, we took the path we wrote in edit1 and saved the result with the imwrite function in the save callback section.

In the callback of the Exit button, close; We closed the gui by typing.

We gave the image from our global variable to the callback function of the Run button, added all our code there and displayed it in axes2.

**GUI RESULTS**

**metin, ekran görüntüsü, diyagram, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin, ekran görüntüsü, diyagram, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin, ekran görüntüsü, diyagram, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**SUMMARY**

The development process of our project initially started with a comprehensive literature research. By examining existing methods in the field of image processing, we determined the most appropriate and effective methods that could meet the requirements of our project. By testing the applications of the algorithms on MATLAB, we selected the ones most suitable for our project, and adopted an approach focused on efficiency and success in this process.

Next, we began the coding process by creating a reference drawing using Paint. While developing our codes, we proceeded by observing the status of the matrices in the system step by step, using the disp command at each stage. We noticed that when we switched to a different drawing in each drawing we were successful in, the sequences returned to zero in some pictures. To solve this problem, we adjusted the parameters in the Canny and Hough algorithms.

Another challenge we encountered was that when grouping points in some drawings, the arrays and matrices were larger than the specified reference value. We developed solutions to correct these errors by adjusting the reference matrix dimensions. Additionally, we optimized the Gaussian filter sizes we determined in the Canny algorithm by enlarging or reducing them in some images. In the Haugh algorithm, we focused on solving the errors that occur due to detecting more than one point in some images when determining the reference point and vertex. When we looked at the overall errors, we felt the need to make some improvements in the system. We analyzed the statistical properties of the drawings and tried to adjust the parameter settings according to these properties. For images with higher edge density, we tried to use a lower threshold value.

We tested the system on more than 50 drawings and successfully achieved the results we wanted on more than 70% of the images we entered. In the last stage of the project, we completed the GUI design and ran the tests via the GUI. We completed our project by successfully completing these processes before the completion date of the project.

To further improve our project, algorithms or various techniques can be used that optimize the parameter settings of the methods we use in the project. For example, using a pre-labeled data set, we can train a machine learning model that learns the appropriate parameter settings for different image types and conditions, or adjust the parameter settings according to a certain performance metric. We may use algorithms or other optimization techniques that iteratively adjust the results to give the best results.