DSAA GROUP PROJECT

GROUP – 11

Members: Vaishnavi N, Syed Jahangir, V Tejkiran, V Sai Rathan, Srilekha N

**DATA COLLECTION**

**Details of the Dataset:**

1. The project has only two databases –

Samples of all handwritten text and digits (55 samples)

Samples of all universal fonts text and digits (1016 samples)

Total samples images = (1016+55) \* (10+26+26) = 66402 images

**SOURCE: -**

MNIST Handwritten Letters Dataset

MNIST Handwritten Digits Dataset

MNIST Font Letters Dataset

MNIST Font Digits Dataset

**DATABASE SIZE AND ATTRIBUTES: -**

(i) It contains 1077-character images of 62 subjects.

(ii) All character images are PNG files and the image resolution of each is 128\*128 px.

(iii) 55 volunteers contributed handwritten character images of 62 characters (small letters, capital letters and numbers).

(iv) The total size of our dataset is 94.9 MB.

**PLANNING**

**ALGORITHM DESIGN: -**

* For Text Extraction, in the pre-processing stage, binarization of image is done. Next we need to detect text regions alone from the image by clipping out unnecessary regions first across each line and then for each line across every character. This helps in getting a crisp outline for each letter.
* After this, we need to segment individual characters one by one and store it.
* After all this, we need to properly resize the characters.
* This process takes the maximum amount of time and hence needs to be slowly and properly implemented.
* After this, feature extraction is done and then Recognition and combining the letters is to be done and then eventually, they must be merged to form valid words and form meaningful sentences.
* Finally, checking its correctness is to be performed. At this point of time, we are ready with our sentences.
* Now, we need to speech synthesize this sentence using System. Speech assembly in MATLAB which will return a voice object and our required voice will thus be generated when the speak function of this object is called.

**PERFORMANCE ANALYSIS: -**

* The proposed system’s accuracy is determined by comparing the final output with the input image.
* It turns out to be 80-90 % accurate in the skeleton level itself without further optimizations.

**COMPLEXITY: -**

* If we are considering an image of size m\*n, then
* To perform horizontal and vertical segmentation it takes 2\*O(m\*n).
* For resizing characters, it takes O(m\*n).
* Therefore, it takes O(m\*n) complexity.

**CODING: -**

* The code snippets are attached below.
* To load input image.

[filename, pathname] = uigetfile('\*','LOAD AN IMAGE');

imagen=imread(fullfile(pathname, filename));

* Binarisation of the image

% Convert to gray scale

if size(imagen,3)==3 %RGB image

imagen=rgb2gray(imagen);

end

% Convert to BW

threshold = graythresh(imagen);

imagen =~im2bw(imagen,threshold);

* Create a .txt file to store the output and open it in write mode

%Make and open \*.txt as file for write

b=find(filename=='.');

pathname = [pathname filename(:,1:b-1) '.TXT'] ;

[FileName, PathName]= uiputfile('\*.txt','Enregister le text sous ...', pathname);

save(fullfile(PathName, FileName));

fid = fopen(fullfile(PathName, FileName), 'wt');

* Load necessary dataset.

% Load templates

load templates

global templates

* Finding the actual character by comparing with our dataset and storing the output.

For each character in every line:

% Call function to resize each char to make comparisons easy.

img\_r=imresize(fc,[42 24]);

% Call function to convert image to text.

letter=read\_letter\_perso(img\_r,num\_letras);

% Letter concatenation

word=[word letter];

* Printing the final string in the file and closing it.

fprintf(fid,word);

fclose(fid);

%Open '\*.txt' file

winopen(fullfile(PathName, FileName))

**PROCESS OF TESTING ANALYSIS: -**

* Initially, all the templates are added to the database and is used for training. Then, Machine is trained on dataset and then when an input image of an English sentence is given, template matching is done. The selected input image will then be compared with all the available character images present in the database and then outputs the character it is closest matched to.

--------------------------------------------------------------X--------------------------------------------------------------