GENERATING OTP USING FINGERPRINTS

In this algorithm , we generate unique OTPs using fingerprints.

The dataset was downloaded from

" http://bias.csr.unibo.it/fvc2000/Downloads/DB3_B.zip ".

Firstly , we input the images from the dataset into a variable **dinfo** which extracts each image information from the dataset using dir command(which takes us to the specified location where the dataset exists).

We loop through dinfo to access each image through its name parameter i.e we store it in a variable

(thisimage = dinfo(K).name)

We use the matlab command "**imread**" for reading the image,

imread(filename) reads a greyscale or color image from the file specified by the string filename , where the string fmt specifies the format of the file. If the file is not in the current directory or in a directory in the **MATLAB** path, specify the full pathname of the location on your system.

Here instead of giving the location we took the image read from dinfo.name

Now , we convert the RGB image read by imread into grayscale image using the function rgb2gray

I = rgb2gray(RGB) converts the truecolor image RGB to the grayscale image I. The rgb2gray function converts RGB images

to grayscale by eliminating the hue and saturation information while retaining the luminance.

Next, we will be detecting the SURFFeatures in the image read

points = detectSURFFeatures(I) returns a SURFPoints object, points, containing information about SURF features detected in the 2-D grayscale input image I. The detectSURFFeaturesfunction implements the Speeded-Up Robust Features (SURF) algorithm to find blob features.

Now,we use extractFeatures function which takes two input parameters – Image and points that were detected earlier using detectSURFFeatures function. extractFeatures function returns features and valid_points of the input image.

[features,validPoints] = extractFeatures(I,points) returns extracted feature vectors, also known as descriptors, and their corresponding locations, from a binary or intensity image.

The function derives the descriptors from pixels surrounding an interest point. The pixels represent and match features specified by a single-point location. Each single-point specifies the center location of a neighborhood. The method you use for descriptor extraction depends on the class of the input points.

We display the grayscale image using figure , imshow and hold on commands in matlab.

Imshow takes image as input parameter.

figure creates a new figure window using default property values. The resulting figure is the current figure.

imshow(I) displays the grayscale image I in a figure.
imshow uses the default display range for the image data type and optimizes figure, axes, and image object properties for image display.

hold on retains plots in the current axes so that new plots added to the axes do not delete existing plots. New plots use the next colors and line styles based on the ColorOrder and LineStyleOrder properties of the axes. MATLAB adjusts axes limits, tick marks, and tick labels to display the full range of data. If axes do not exist, then the hold command creates them.

We extract the size of features vector using size function in matlab.

sz = **size(A)** returns a row vector whose elements contain the length of the corresponding dimension of A. For example, if A is a 3-by-4 matrix, then size(A) returns the vector [3 4]. The length of sz is ndims(A). If A is a table or timetable, then size(A) returns a two-element row vector consisting of the number of rows and the number of table variables.

Since we get a 2 x 1 vector when we use size function , we access the element of the 2 x 1 vector which gives the count of number of rows of features vector.For example , let A be a 2x1 vector , the first element can be accessed using A(1).

Here , we accessed the row count of features vector using

siz_feat(1)

Next , we generate a random number ranging from 1 to siz_feat(1).This is achieved through **randperm** function in matlab.This function doesn't repeat a random number in a given range.

p = **randperm(n)** returns a row vector containing a random permutation of the integers from 1 to n inclusive.

For example , randperm(6) might be the vector [3 2 6 4 1 5].

Now , we take the first element of the vector returned by randperm function and store it in a variable s.

Next, we store first "s-1" x-coordinates, y-coordinates of the valid points in variables X and Y respectively.

X=valid_points.Location(1:s-1,1);

Y=valid_points.Location(1:s-1,2);

Next,we find the angle between two consecutive valid_points in degrees using "**atan**" function and store it in a variable "a". We find the length between two consecutive points and store it in a variable "l",using "**norm**" function. In order to obtain an integer we will round it off using "**round**" function(for both a and l).

n = norm(v) returns the Euclidean norm of vector v. This norm is also called the 2-norm, vector magnitude, or Euclidean length.

Now ,we apply bitxor between angle and length which we calculated in the above step.The inputs given in bitxor are explicitly converted from “double” to “int64”.

k3=bitxor(int64(l),int64(a),'int64');

C **= bitxor**(A,B) returns the bit-wise XOR of A and B.

Input values, specified as scalars, vectors, matrices, or multidimensional arrays. Inputs A and B must either be the same size or have sizes that are compatible (for example, A is an M-by-N matrix and B is a scalar or 1-by-N row vector). For more information, see Compatible Array Sizes for Basic Operations. A and B also must be the same data type unless one is a scalar double.

->If A and B are double arrays, and assumedtype is not specified, then MATLAB® treats A and B as unsigned 64-bit integers.

->If assumedtype is specified, then all elements in A and B must have integer values within the range of assumedtype.

Data Types: double | logical | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64

The value obtained frm bitxor is added to a variable ”f” every time the loop executes. Also we add the value at “c”th row of features vector to a variable “sum”(where “c” is the iterator which ranges from 1 to “s-2”).

f=f+k3;

sum=sum+features(c);

Next we multiply the variables “sum” and “f” obtained above and convert them explicitly to double datatype. Now we convert this double value into int16 datatype using typecast.We now convert this result into binary using dec2bin function in matlab.

str = dec2bin(d) returns the binary representation of d as a character vector. d must be a nonnegative integer. If d is greater than the value returned by flintmax, then dec2bin might not return an exact representation of d.

str = dec2bin(d,n) produces a binary representation with at least n bits.

The output of dec2bin is independent of the endian settings of the computer you are using.

Examples

Decimal 23 converts to binary 010111:

dec2bin(23)

ans =10111

Next , we consider the binary string obtained from dec2bin function as 8 bit characters(**we convert the string into 8 bit characters using unit8 function**) and pass the result to **sha256**

hashing algorithm,which coverts it into a vector of decimal values.

sha256hasher = System.Security.Cryptography.SHA256Managed;

sha256 = uint8(sha256hasher.ComputeHash(uint8(val)));

A cryptographic **hash** is an **algorithm** that takes an input and turns it into an output of a fixe d size. It looks like a mix up of numbers and letters. There are many types of cryp tographic hashes. Bitcoin, for example, uses a hashing **algorithm** called **SHA**-**256**. SHA256 is a hashing function, not an encryption function. Secondly, since SHA256 is not an encryption function, it cannot be decrypted. What you mean is probably reversing it. In that case, SHA256 cannot be reversed because it's a one-way function.

Now,we convert the decimal-valued vector returned by hashing algorithm into hexadecimal-valued character vector using dec2hex function.

dec2hex

str = dec2hex(d) returns the hexadecimal representation of d as a character vector. d must be a nonnegative integer. If d is an integer greater than the value returned by flintmax, then dec2hex might not return an exact representation. MATLAB® converts noninteger inputs, such as those of class double or char, to their integer equivalents before converting to hexadecimal.

Example

To convert decimal 1023 to hexadecimal,

dec2hex(1023)

ans =3FF

We store the hexdecimal character vector returned by dec2hex function in a variable “x”. From “x” , we form an OTP of length 6.

The first character of the OTP is the first character of first element of x.

Second character of OTP is the first character of last element of x.

Third character of OTP is the first character of center element of x.

Fourth character of OTP is the second character of first element of x.

Fifth character of OTP is the second character of last element of x.

Sixth character of OTP is the second character of center element of x.

Now , we write this OTP result into an excel file along with the name of the input image processed .To do this , we use xlswrite function.

xlswrite(filename,A) writes matrix A to the first worksheet in the Microsoft® Excel® spreadsheet workbook filename starting at cell A1.

xlswrite(filename,A,sheet) writes to the specified worksheet.

xlswrite(filename,A,xlRange) writes to the rectangular region specified by xlRange in the first worksheet of the workbook. Use Excel range syntax, such as 'A1:C3'.

xlswrite(filename,A,sheet,xlRange) writes to the specified worksheet and range.

Here , we input xlswrite function with filepath , vector whose 1st element is image name and the 2nd element is the OTP we obtained from the image , spredsheet number , cellname in spreadsheet to store the vector(i.e [image_name OTP]).

We change the cellname of spreadsheet by incrementing a variable offset , which was initially initialized to 1.(i.e our data gets stored from A1 cell in the spreadsheet).

str **= sprintf(formatSpec,A1,...,An)** formats the data in arrays A1,...,An using the formatting operators specified by formatSpec and returns the resulting text in str. The sprintf function formats the values in A1,...,An in column order. If formatSpec is a string, then so is the output str. Otherwise, str is a character vector.

Example

formatSpec = 'The array is %dx%d.';

```
A1 = 2;A2 = 3;
str = sprintf(formatSpec,A1,A2)
str = 'The array is 2x3.'
```