IMAGE PROCESSING PROJECT REPORT

Prepared by

Group 2

Hand Gesture Recognition System

Overview

Findings published in the world health statistics 2015 by the World Health Organization state that as of 2015 about 5%, approximately 70 million, of the world's population being deaf and mute. Most of them are using Sign Language to communicate with each other.

Problem

Sign languages (also known as signed languages) are languages that use the visual-manual modality to convey meaning. Sign languages are expressed through manual articulations in combination with non-manual elements and are used to communicate among the deaf and mute community. The main drawback of sign language is people who don't have hearing issues haven't any idea about it. That means there is no way for deaf people to communicate their idea when the other party doesn’t know about sign language.

Solution

First implemented the algorithms to identify and filter images to a readable format for **MATLAB**. Then implemented a file system using the alphabet of the sign language which can be used to identify the inputs. After that implemented the user interface that the user can input a photo of a hand gesture which will be understood by the software.

We took this solution because this is a much faster way to understand hand gestures. Also, the person who needs to interact with the deaf person does not need to have any knowledge about sign language. Anyone with this system also can learn sign language by observing the images. So we thought this system would be a great opportunity for people who are trying to interact with deaf people and putting themselves in an uncomfortable moment when they try to understand their signs.

The hand gesture recognition system is mainly for people who are interacting with deaf people. This program was created to understand the hand sign language of the deaf person and give a meaningful output to the user.

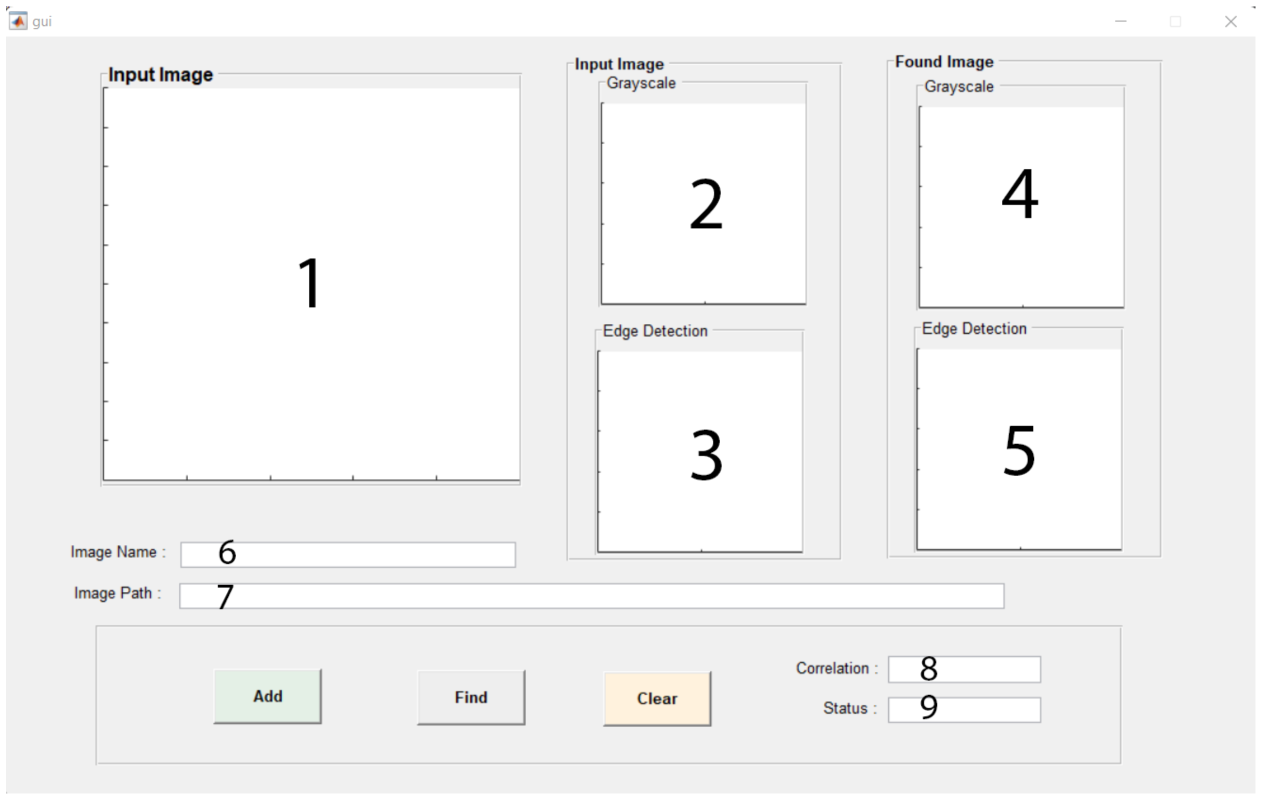
Product Overview

Product Details

Developed Platform	: Matlab 2015
Input	: JPG
Output	: JPG
Size	: 84KB

Getting started with the product

Basic Process of the system: Identify and show the matched output of the given picture



Explaining the Functions of the Buttons

1. Add Button - Opens a window to browse for a picture
2. Find Button - Start the process of finding the matching picture and display to the user
3. Clear Button - Clear the current data from the windows

Explaining the areas of the window

1 : Display the inputted image
2 : Display the Grayscale view of the inputted image
3 : Display the Edge detected view of the inputted image
4 : Display the Grayscale view of the matching image
5 : Display the Edge detected view of the matching image
6 : Display the inputted image name
7 : Display the inputted image path
8 : Display the correlation value of the inputted image and the found matching image
9 : Display the status (Found/Not Found)

Image with Data :

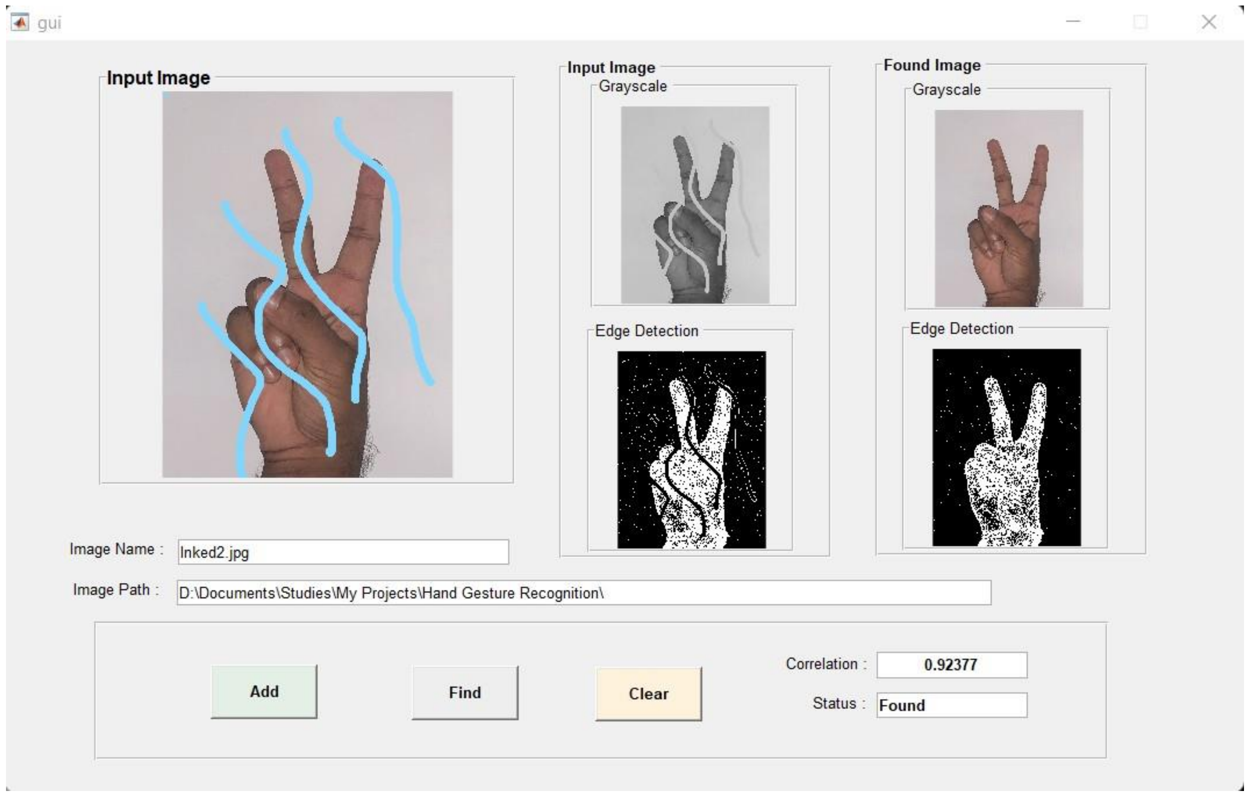


Image Name : Inked2.jpg
Image Path : D:\Documents\Studies\My Projects\Hand Gesture Recognition\
Correlation : 0.92377
Status : Found

Used Algorithms to detect image

2-D correlation coefficient method

Syntax

r = corr2(A , B)

r = corr2(gpuarrayA , gpuarrayB)

Description

r = corr2(A,B) returns the correlation coefficient r between A and B, where A and B are matrices or vectors of the same size. r is a scalar double.

r = corr2(gpuarrayA,gpuarrayB) performs the operation on a GPU. The input images are 2-D gpuArrays of the same size. r is a scalar doublegpuArray. This syntax requires the Parallel Computing Toolbox.

Code:

```
for i = 1:6
   c{i}=imread(sprintf('%d.JPG',i));
   foundImage = medfilt2(rgb2gray(c{i}));
   r = corr2(inputImage,foundImage);
   if r>0.8500
      set(handles.edit3,'String','Found');
      x=x+1;
      dd = num2str(r);
      set(handles.edit4,'String',dd);
      imshow(c{i});

      %Displaying Edge Detected Image
      im3 = edge(foundImage,'canny');
      %dilation
      im4 = bwmorph(im3,'dilate',3);
      axes(handles.a5);
      imshow(im4);
```

```
          break;
        end
      end
```

Matlab Code

```
%Created By Group 2

function varargout = gui(varargin)
% GUI MATLAB code for gui.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 13-Dec-2021 18:15:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
                   'gui_OutputFcn', @gui_OutputFcn, ...
```

```
                'gui_LayoutFcn', [] , ...
                'gui_Callback', []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

% Choose default command line output for gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes on button press in add.
function add_Callback(hObject, eventdata, handles)
[rawname, rawpath] = uigetfile({'*.jpg'},'Select Image Data');
global fullname
fullname = [rawpath rawname];
```

```
set(handles.edit1,'String',rawname);
set(handles.edit2,'String',rawpath);

inputI = imread(fullname);
axes(handles.a1);
imshow(inputI);

% --- Executes on button press in find.
function find_Callback(hObject, eventdata, handles)

%clearing the fields before running again
cla(handles.a2);
cla(handles.a3);
cla(handles.a4);
cla(handles.a5);
set(handles.edit3,'String','');
set(handles.edit4,'String','');

%Declaring and Intializing Variables
global fullname
RGB = imread(fullname);

%Converting RGB Input Image to Grayscale
grayI = rgb2gray(RGB);
axes(handles.a2);
imshow(grayI);

%Displaying Edge Detected for the input Image
im1 = edge(grayI,'canny');
im2 = bwmorph(im1,'dilate',3);
axes(handles.a4);
imshow(im2);

%Converting Grayscale Input Image to
inputImage = medfilt2(grayI);

%Creating a cell array
c = cell(1,6);

x = 0;
axes(handles.a3);
```

```
for i = 1:6
   c{i}=imread(sprintf('%d.JPG',i));
   foundImage = medfilt2(rgb2gray(c{i}));
   r = corr2(inputImage,foundImage);
   if r>0.8500
      set(handles.edit3,'String','Found');
      x=x+1;
      dd = num2str(r);
      set(handles.edit4,'String',dd);
      imshow(c{i});

      %Displaying Edge Detected Image
      im3 = edge(foundImage,'canny');
      %dilation
      im4 = bwmorph(im3,'dilate',3);
      axes(handles.a5);
      imshow(im4);
      break;
   end
end

if x == 0
   set(handles.edit3,'String','Not Found');
end

% --- Executes on button press in clear.
function clear_Callback(hObject, eventdata, handles)
cla(handles.a1);
cla(handles.a2);
cla(handles.a3);
cla(handles.a4);
cla(handles.a5);
set(handles.edit1,'String','');
set(handles.edit2,'String','');
set(handles.edit3,'String','');
set(handles.edit4,'String','');

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Conclusion

This is a great opportunity for deaf/mute people to communicate with normal people. And normal people are also able to communicate with deaf people by using this application. Our main goal is to improve the relationship between normal people and deaf people. This is just a part of the program that we planned to implement in the future.

We would like to improve our current algorithms and give a more accurate and efficient system. We also like to add a real-time hand gesture detection system. That the practical issue of the current system will be solved.

Group Members

- Herath H.M.S.D.B
- Perera M.M.B
- Dhananjaya K.A.D.B
- Perera W.K.D.D