Chapter 6

Implementation

6.1 Algorithm

Step 1: Stained and magnified human blood smear image is fed to the program as input.

Step 2: Input image is converted to grayscale and is enhanced by performing histogram stretching.

Step 3: The void spaces within the cells of the enhanced image are filled using the imfill() command. Use of imfill() would require the image to be complemented before and after the operation since the function only fills white into blacks spaces bounded by white boundaries and not vice-versa.

Step 4: The resulting image is then converted into a binary image.

Step 4: The image is complemented and small unwanted spots are discarded from the image using 'bwareaopen', which removes all the objects in the diagram containing fewer than the number of pixels mentioned in the threshold level (here, threshold level= 50). The image is then complemented back to its original form for further processing.

Step 5: The cells/objects intersecting the borders of the image would produce inaccurate results. Therefore, such cells are discarded using imclearborder().

Step 6: Individual cells are detected by using bwconncomp() command in Matlab. bwconncomp() returns a structure of connected components in the image passed to it. This structure is then passed to regionprops() to obtain the area and perimeter of every cell.

Step 7: The number of valid cells are computed by considering only the cells falling within the quartile range.

Step 8: The metric value for every valid cell is computed and stored. Metric value is computed using the formula $4*PI*area/(perimeter)^2$.

Step 9: The valid cells are further classified as normal and abnormal cells depending on the metric value corresponding to the cells. A threshold metric value is decided (usually above 0.75), above which a cell is classified as normal.

Step 10: A threshold is decided depending on which a decision is made whether the individual is diagnosed with the disease or not. If the number of abnormal cells is equal to or more than the threshold, a positive SCA result is displayed.

6.2 Working of the project

ASCAD.fig

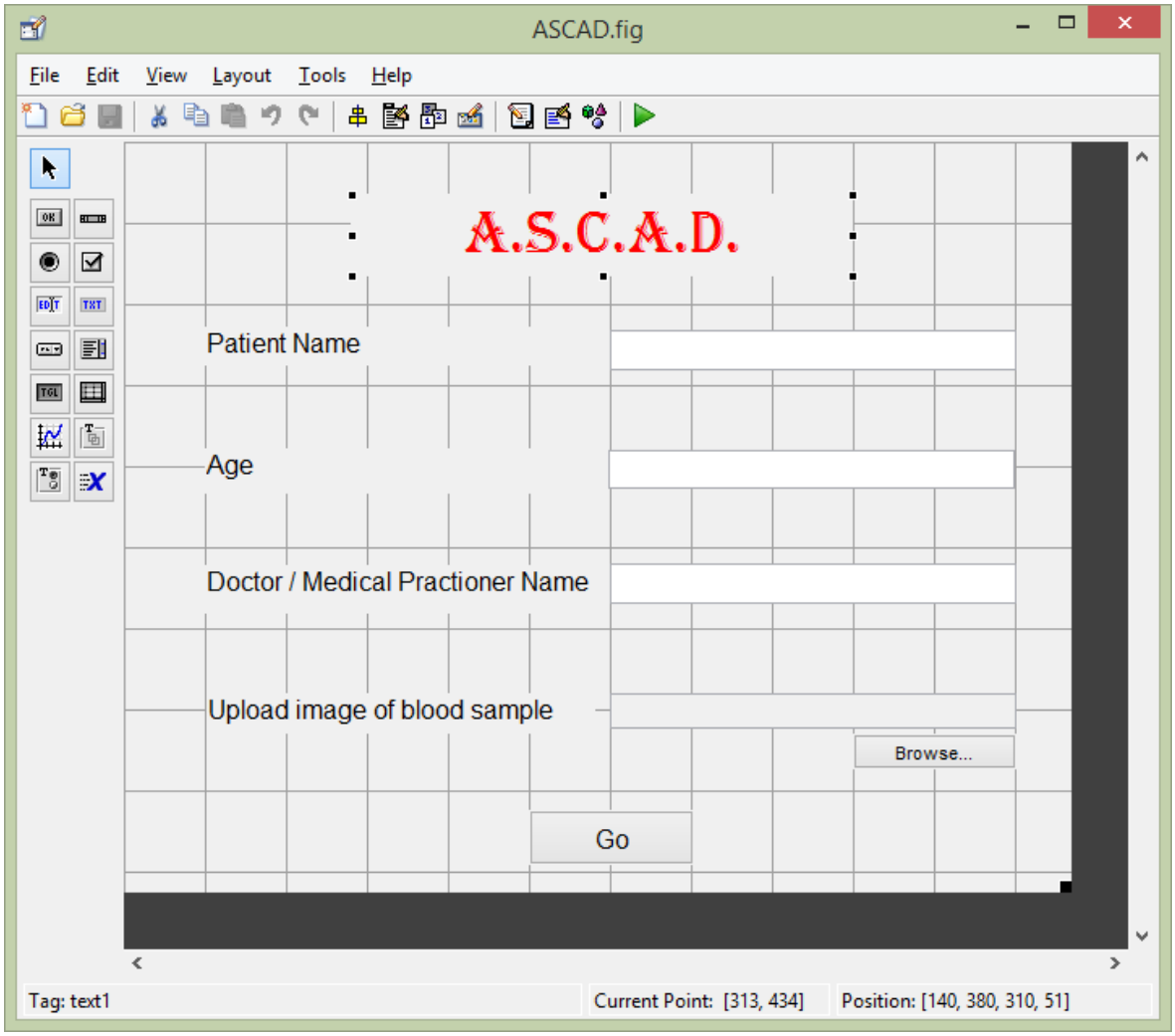

Fig. 6.2.1 GUI layout

ASCAD.m

```
function varargout = ASCAD(varargin)
% ASCAD MATLAB code for ASCAD.fig
%      ASCAD, by itself, creates a new ASCAD or raises the existing
%      singleton*.
%
%      H = ASCAD returns the handle to a new ASCAD or the handle to
%      the existing singleton*.
%
%      ASCAD('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ASCAD.M with the given input arguments.
%
%      ASCAD('Property','Value',...) creates a new ASCAD or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ASCAD_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ASCAD_OpeningFcn via varargin.

% Last Modified by GUIDE v2.5 07-Apr-2015 16:20:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ASCAD_OpeningFcn, ...
                   'gui_OutputFcn',  @ASCAD_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 ASCAD is made visible.
function ASCAD_OpeningFcn(hObject, eventdata, handles, varargin)
% 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 ASCAD (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);
```

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


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



function P_name_Callback(hObject, eventdata, handles)


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



function P_age_Callback(hObject, eventdata, handles)


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



function D_name_Callback(hObject, eventdata, handles)


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



function img_path_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function img_path_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
handles.output = hObject;
[fn pn] = uigetfile('*.jpg','select dicom file');
complete = strcat(pn,fn);
set(handles.img_path,'string',complete);

% --- Executes on button press in Go.
function Go_Callback(hObject, eventdata, handles)
P= get(handles.P_name, 'String');
A= str2num(get(handles.P_age, 'String'));
D= get(handles.D_name, 'String');
IP= get(handles.img_path, 'String');

if isempty(P)     % checking that the patient name field is not left blank
    errordlg('Please enter the Patient Name.', 'Error');
else if isempty(A)    % checking that the patient age is not left blank
        errordlg('Please enter the age of the patient.', 'Error');
else if A<0     % checking if the user has entered a negative age
        errordlg('Please enter a valid age of the patient.', 'Error');
else if isempty(D);
        errordlg('Please enter the name of the doctor or medical practioner
who advised the test. Enter N.A. in case of self test.', 'Error');
else if isempty(IP)
        errordlg('Image path empty. Please browse to the input
image.','Error');
    else
        Code(get(handles.P_name, 'String'), get(handles.P_age, 'String'),
get(handles.D_name, 'String'), get(handles.img_path, 'String'));
    end
    end
    end
    end
end
```

Code.m

```
function Code(P, A, D, IP)
warning off;

P_name= P;     %input('Enter Patient name : ','s');
P_age= A;     %input('Enter Patient age : ');
D_name= D;     %input('Enter Doctor (medical practioner) name : ','s');
%img_input = input('Enter the image index : ');
img_path= IP;     %strcat('C:\AAA\Img\',int2str(img_input),'.jpg');
img= imread(img_path);
```

```
%figure, imshow(img);

img_grayS = rgb2gray(img);
%figure, imshow(img_grayS);

%histogram stretching
bins = linspace(0,255,256);
H = hist(img_grayS(:), bins);
H(H==0) = eps(sum(H));
cdf = [0,cumsum(H)/sum(H)];    %cumulative distribution function
pct= 0.05;    %percent of pixel values to ignore
h_low = interp1(cdf, [0,bins], pct);
h_high = interp1(cdf, [0,bins], 1-pct);
stretchedImg= uint8((double(img_grayS)-h_low)/(h_high-h_low) * 255);
%img_grayS= uint8(imadjust(img_grayS, stretchlim(img_grayS), []));
%histogram stretching using inbuilt function
%figure, imshow(stretchedImg);

BW_c= imcomplement(stretchedImg);
BW_filled= imfill(BW_c,4, 'holes');
BW_refined= imcomplement(BW_filled);
%figure, imshow(BW_refined);

%Gaussian filter
myfilter = fspecial('gaussian',[3 3],16);
filteredImg = imfilter(BW_refined, myfilter, 'replicate');


img_binary= im2bw(BW_refined,graythresh(filteredImg));   % converting the
enhanced image to binary
%figure, imshow(img_binary);

img_binary_c= imcomplement(img_binary);    %bwareaopen and bwconncomp works
only on bright pixels
refined_img= bwareaopen(img_binary_c,200, 4);    %removing unwanted spots
from the image having a maximum pixel density of 200 pixels
%figure, imshow(refined_img);

%EROSION (EXPERIMENTAL)
se = strel('disk',2);
erodedBW = imerode(refined_img,se);    %Shrinks each 'disk' shaped object in
the image
%figure, imshow(erodedBW);

img_binary1 = imclearborder(erodedBW); %remove border objects
%figure, imshow(img_binary1);

CC= bwconncomp(img_binary1, 4);    % keeping connectivity 4 to ignore
diagonal connection
CA= regionprops(CC, 'Area');    %returns the areas of all objects in the
image identified by bwconncomp()
CP= regionprops(CC, 'Perimeter');    %returns the perimeters of all objects
in the image identified by bwconncomp()
img_binary2= imcomplement(img_binary1);
```

```
areas= cell(CC.NumObjects,1);    %storing the area of individual components
from CA.Area
                                 %CA.Area is not feasible to use for
computations

for i=1: CC.NumObjects
    areas{i,1}= CA(i,1).Area;
end
areas= cell2mat(areas);    %converting the cell to array

iqr= quantile(areas,0.75)-quantile(areas,0.25);      %inter quartile range

average=iqr*1.5;    %going by the definition of an outlier

%counting the number of valid and invalid cels in the image (only considering
the objects with areas within the inter quartile range)
validCells=0; invalidCells=0;
for i=1:CC.NumObjects
    if CA(i,1).Area >= quantile(areas,0.25)-average && CA(i,1).Area <=
quantile(areas,0.75)+ average
        validCells= validCells+1;
    else
        invalidCells= invalidCells+1;
    end
end

%calculating the metric value for each valid cell
metric= cell(validCells ,1);
k=1;
for i=1:CC.NumObjects
    if CA(i,1).Area >= quantile(areas,0.25)-average && CA(i,1).Area <=
quantile(areas,0.75)+ average
        metric{k,1} = (4*pi*CA(i,1).Area)/(CP(i,1).Perimeter *
CP(i,1).Perimeter);
        k=k+1;
    end
end

normal=0; abnormal=0;

for i=1:validCells
   if metric{i,1}>=0.75    %all cells with a metric value above 0.75 are
considered normal, as they prove to be more circular
       normal=normal+1;
   else
       abnormal=abnormal+1;
   end

end

threshold=0.1*validCells;    % 10 percent of the number of valid cells
(Experimental)
                             % Reference - http://goo.gl/E4NCam
```

```
%printing the report
disp(sprintf('\n\nReport :-'));
disp(sprintf('Patient name : %s', P_name));
disp(sprintf('Patient age : %s', P_age));
disp(sprintf('Doctor / Medical practioner name : %s', D_name));
disp(sprintf('Image Path : %s', img_path));
disp(sprintf('\nTotal number of objects detected\t=  %d',CC.NumObjects));
disp(sprintf('Number of valid cells\t\t\t\t=  %d',validCells));
disp(sprintf('Number of normal cells\t\t\t\t=  %d',normal));
disp(sprintf('Number of abnormal cells\t\t\t=  %d',abnormal));
disp(sprintf('Threshold \t\t\t\t\t\t\t=  %f',threshold));
if abnormal>threshold
    disp(sprintf('\nResult : \t\tYou are diagnosed with Sickle Cell
Anaemia.\n'));
    msgbox(sprintf(' Patient name : %s\n Patient age : %s\n Doctor / Medical
practioner name : %s\n Image Path : %s\n\n Total number of objects
detected\t=  %d\n Number of valid cells\t\t\t\t=  %d\n Number of normal
cells\t\t\t\t=  %d\n Number of abnormal cells\t\t\t=  %d\n Threshold
\t\t\t\t\t\t\t=  %f\n\n Result : \t\tYou are diagnosed with Sickle Cell
Anaemia.\n', P_name, P_age, D_name, img_path, CC.NumObjects, validCells,
normal, abnormal, threshold),'Report- Positive');
else
    disp(sprintf('\nResult : \t\tÝou are safe.\n'));
    msgbox(sprintf(' Patient name : %s\n Patient age : %s\n Doctor / Medical
practioner name : %s\n Image Path : %s\n\n Total number of objects
detected\t=  %d\n Number of valid cells\t\t\t\t=  %d\n Number of normal
cells\t\t\t\t=  %d\n Number of abnormal cells\t\t\t=  %d\n Threshold
\t\t\t\t\t\t\t=  %f\n\n Result : \t\tYou are safe.\n', P_name, P_age, D_name,
img_path, CC.NumObjects, validCells, normal, abnormal, threshold),'Report-
Negative');
end

end    %function Code end
```