

**Design/Practical Experience [EEN3010]**  
**Department of Electrical Engineering**  
**Final Report**

**Academic Year: 2021-22**

**Semester: Summer**

Date of Submission of Report: 07-07-2022

1.Name of the Student: **Rohan Nehra**

2.Roll Number: **B19EE071**

3.Title of the Project: **Design of a MATLAB based tool for droplet sizing using image processing**

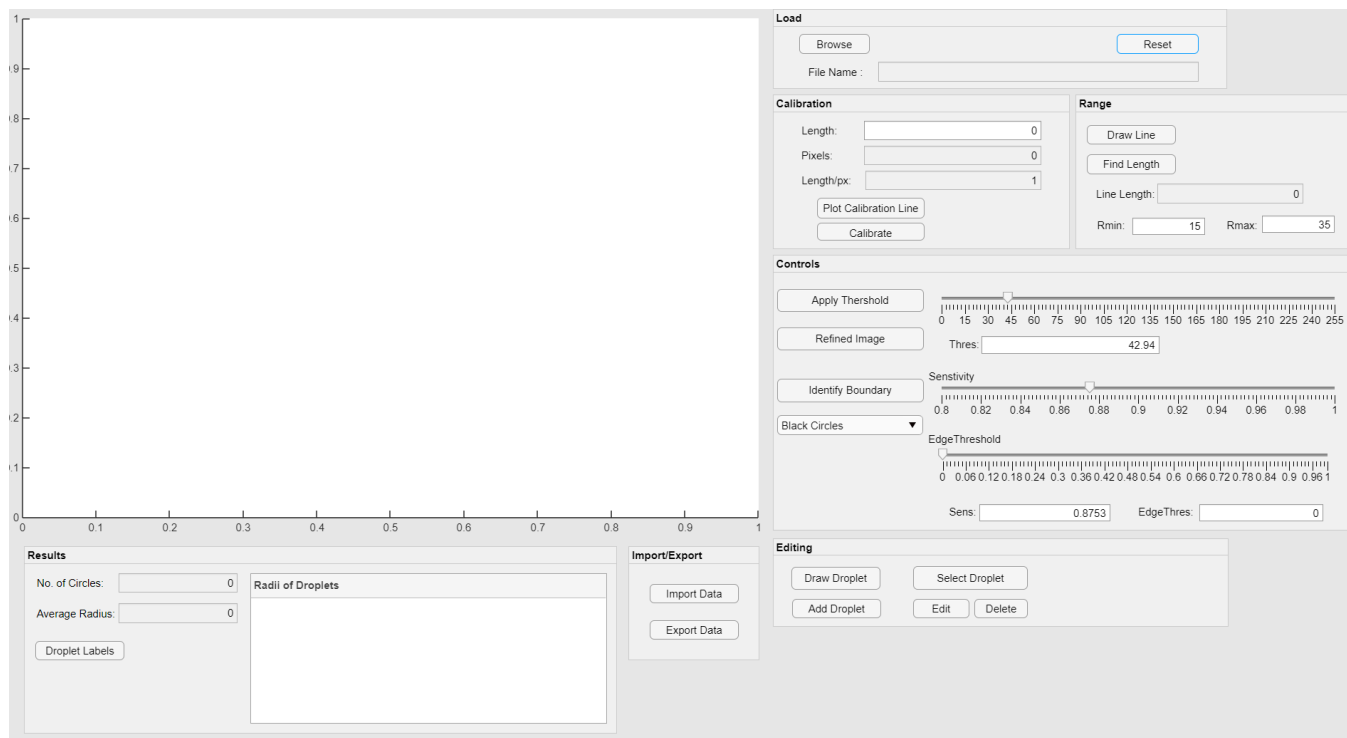
4.Project Category: **Category 3**

5.Supervisor/Mentor Name: **Dr. Sudipto Mukhopadhyay**

6.Targeted Deliverables: Develop an interactive tool using MATLAB codes that identifies the droplets and their sizes from the image taken.

7.Work Done:

**Abstract:** An application is designed that can measure the average radius of the droplets of a spray chamber image. Droplet size is an essential parameter in spray modeling. The tool detects the circles from the provided image and then measures the average radius of the circles as droplets are almost circular. This is the GUI of the application.



## Working:

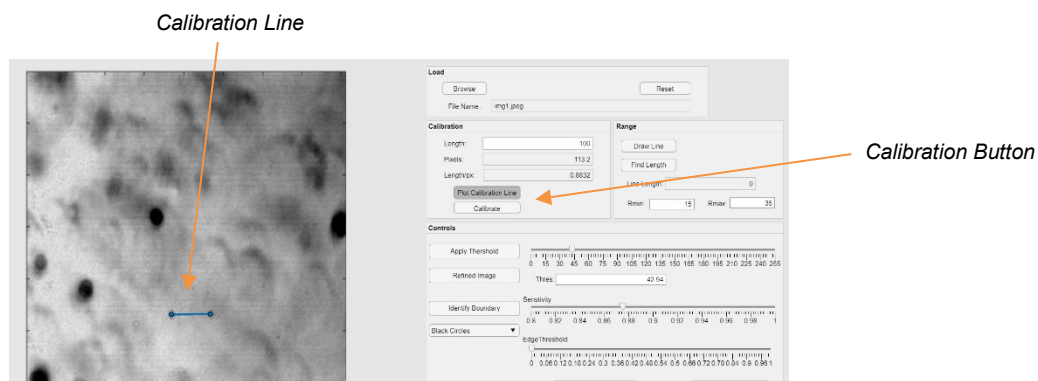
### I. Import Image:

The image can be imported from the device using the 'Browse' button. This is the MATLAB code:

```
function BrowseButtonPushed(app, event)
    [app.file,app.path] = uigetfile({'*.png;*.jpg','Images (*.png,*.jpg)';'*.*', 'ALL Files (*.*)'});
    if isequal(app.file,0)
        return
    end
    cla(app.UIAxes,"reset");
    [app.img,map] = imread(fullfile(app.path,app.file));
    app.FileNameEditField.Value = app.file;
    if isempty(map)
        app.img = rgb2gray(app.img);
    else
        app.img = ind2gray(app.img,map);
    end
    app.thimg = app.img;
    imshow(app.img,'parent',app.UIAxes);
    app.centers(:) = [];
    app.radii(:) = [];
    showcircles(app)
    app.ApplyThersholdButton.Value = 0;
    app.IdentifyBoundaryButton.Value = 0;
    app.DrawLineButton.Value = 0;
    app.DrawDropLetButton.Value = 0;
    app.SelectDropLetButton.Value = 0;
    app.LengthEditField.Value = 0;
    app.PixelsEditField.Value = 0;
end
```

### II. Calibration of the image:

The calibration is required so that the radius can be measured in the desired unit. Every image has a reference line whose length is known. We can draw a calibration line over that reference line and get length per pixel value and then calibrate the image.



```

function PlotCalibrationLineButtonValueChanged(app, event)
    if ~isempty(app.thimg)
        value = app.PlotCalibrationLineButton.Value;
        if value == 1
            app.clbr = drawline("Deletable",true,"Visible",1,"Parent",app.UIAxes);
        elseif value == 0
            app.clbr.Visible=0;
        end
    else
        app.PlotCalibrationLineButton.Value = 0;
    end
end

function CalibrateButtonPushed(app, event)
    if app.PlotCalibrationLineButton.Value == 1
        pos = app.clbr.Position;
        delete(app.clbr);
        diffPos = diff(pos);
        app.PixelsEditField.Value = hypot(diffPos(1),diffPos(2));
        if app.LengthEditField.Value>0
            app.LengthpxEditField.Value = app.LengthEditField.Value/app.PixelsEditField.Value;
        else
            app.LengthpxEditField.Value = 1;
        end
        app.Lpx = app.LengthpxEditField.Value;
        app.PlotCalibrationLineButton.Value = 0;
        showmeasurements(app);
    end
end

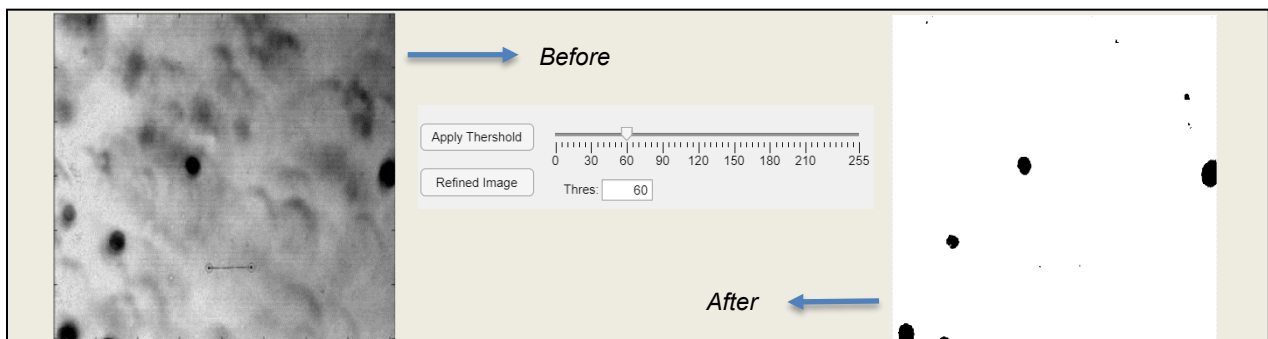
function LengthpxEditFieldValueChanged(app, event)
    value = app.LengthpxEditField.Value;

end

```

### III. Background Removal:

To remove the background, thresholding is used. There is a 'Apply Threshold' button that can apply a certain threshold value to the image. The value can be manually changed using the slider.



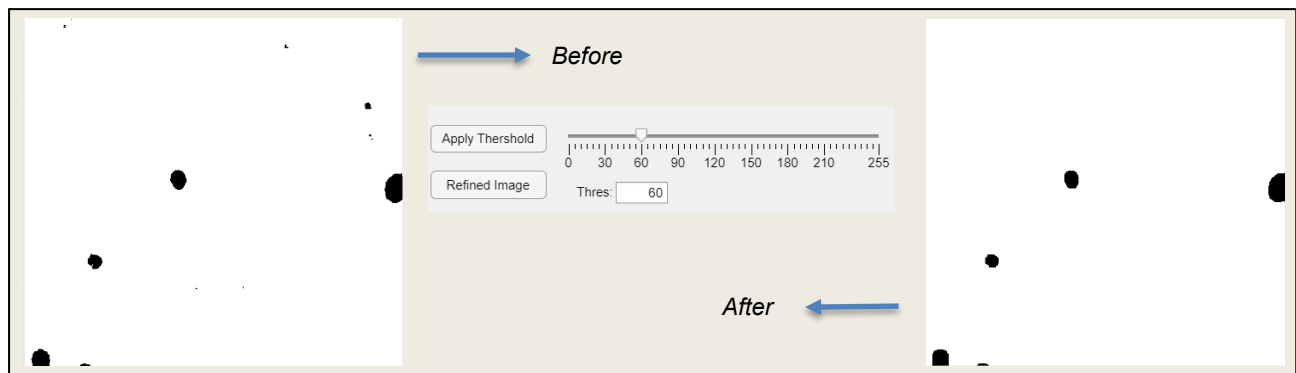
```

function ApplyThersholdButtonValueChanged(app, event)
    value = app.ApplyThersholdButton.Value;
    app.DrawLineButton.Value = 0;
    app.DrawDropletButton.Value = 0;
    app.SelectDropletButton.Value = 0;
    if value ==1
        if ~isempty(app.thimg)
            if app.IdentifyBoundaryButton.Value==1
                app.thimg = app.img > app.ThresEditField.Value;
                [app.centers,app.radii] = imfindcircles(app.thimg,[round(app.RminEditField.Value/app.Lpx)
round(app.RmaxEditField.Value/app.Lpx)],Sensitivity=app.SensEditField.Value,ObjectPolarity =
app.objp,EdgeThreshold=app.EdgeThresEditField.Value);    %,Method='twostage'
                imshow(app.thimg, 'Parent', app.UIAxes);
                showcircles(app)
                showmeasurements(app)
            else
                app.thimg = app.img > app.ThresEditField.Value;
                imshow(app.thimg, 'Parent', app.UIAxes);
            end
        else
            app.ApplyThersholdButton.Value = 0;
        end
    elseif value == 0
        if app.IdentifyBoundaryButton.Value==1
            app.thimg = app.img;
            imshow(app.thimg, 'Parent', app.UIAxes);
            showcircles(app)
            showmeasurements(app)
        else
            app.thimg = app.img;
            imshow(app.thimg, 'Parent', app.UIAxes);
        end
    end
end
end

```

#### IV. Image Refining:

After thresholding, there are some grains in the image which may be detected by the circle detection algorithm so image refining is used to get rid of these grains and make the droplets more circular.



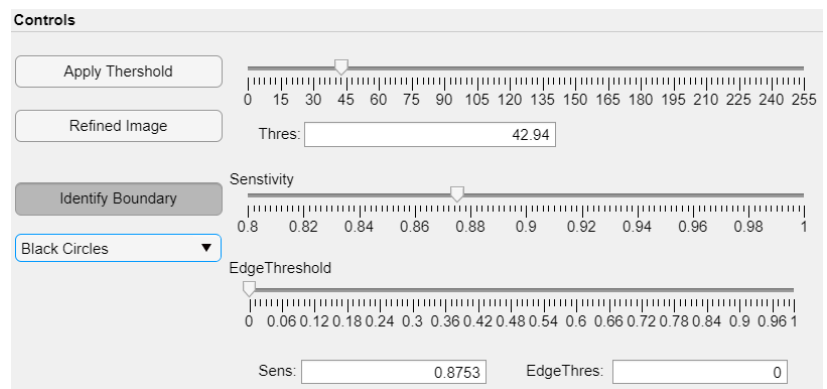
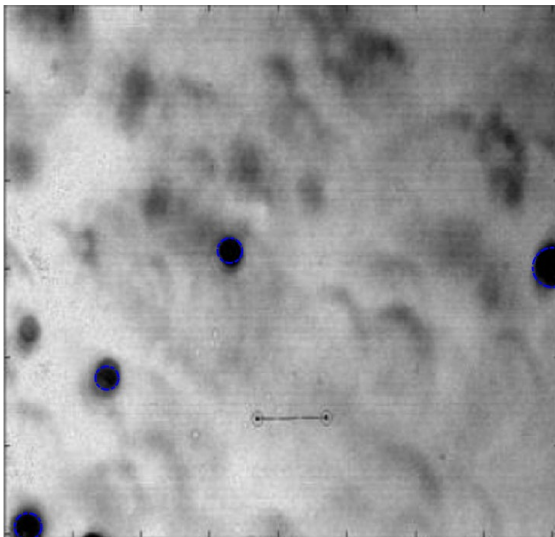
```

function RefinedImageButtonPushed(app, event)
    if app.ApplyThersholdButton.Value == 1
        sel = strel('disk',round(app.RminEditField.Value/2),6);
        if app.IdentifyBoundaryButton.Value==1
            app.thimg = imclose(imopen(app.thimg,sel),sel);
            [app.centers,app.radii] = imfindcircles(app.thimg,[round(app.RminEditField.Value/app.Lpx)
round(app.RmaxEditField.Value/app.Lpx)],Sensitivity=app.SensEditField.Value,ObjectPolarity =
app.objp,EdgeThreshold=app.EdgeThresEditField.Value);    %,Method='twostage'
            imshow(app.thimg, 'Parent', app.UIAxes);
            showcircles(app)
            showmeasurements(app)
        else
            app.thimg = imclose(imopen(app.thimg,sel),sel);
            imshow(app.thimg, 'Parent', app.UIAxes);
        end
    end
end
end

```

## V. Droplet Detection:

The droplets are circular in shape so an algorithm based on circle detection is used. We have to provide the range of radii for circle detection. This range can be calculated using the 'Draw Line' button. We can then click on the 'Identify Boundary' button that we can see the circle drawn over the droplets. The 'sensitivity' and 'edge threshold' sliders can be used to manually change the parameters for the circle detection algorithm. In this image there are black droplets but other images may have white droplets so a button is also provided that can be set to bright or dark based on the color of the circles.



```

function DrawLineButtonValueChanged(app, event)
    if ~isempty(app.thimg)
        value = app.DrawLineButton.Value;
        if value == 1
            app.drl = drawline("Deletable",true,"Visible",1,"Parent",app.UIAxes);
        elseif value == 0
            app.drl.Visible=0;
        end
    else
        app.DrawLineButton.Value = 0;
    end
end

function IdentifyBoundaryButtonValueChanged(app, event)
    value = app.IdentifyBoundaryButton.Value;
    if value==1
        if ~isempty(app.thimg)
            [app.centers,app.radii] = imfindcircles(app.thimg,[round(app.RminEditField.Value/app.Lpx)
round(app.RmaxEditField.Value/app.Lpx)],Sensitivity=app.SensEditField.Value,ObjectPolarity =
app.objp,EdgeThreshold=app.EdgeThresEditField.Value);    %,Method='twostage'
            showcircles(app)
            showmeasurements(app);
        else
            app.IdentifyBoundaryButton.Value = 0;
        end
    elseif value == 0
        app.DrawLineButton.Value = 0;
        app.DrawDropLetButton.Value = 0;
        app.SelectDropLetButton.Value = 0;
        imshow(app.thimg,'parent',app.UIAxes);
        app.centers(:) = [];
        app.radii(:) = [];
        showcircles(app)
        showmeasurements(app)
    end
end

% Value changing function: SensitivitySlider
function SensitivitySliderValueChanging(app, event)
    changingValue = event.Value;
    app.SensEditField.Value = changingValue;
    if app.IdentifyBoundaryButton.Value == 1
        if ~isempty(app.thimg)
            [app.centers,app.radii] = imfindcircles(app.thimg,[round(app.RminEditField.Value/app.Lpx)
round(app.RmaxEditField.Value/app.Lpx)],Sensitivity=app.SensEditField.Value,ObjectPolarity =
app.objp,EdgeThreshold=app.EdgeThresEditField.Value);    %,Method='twostage'
            app.h.Visible = 0;
            showcircles(app);
            showmeasurements(app);
        else
            app.IdentifyBoundaryButton.Value = 0;
        end
    end
end
end

```

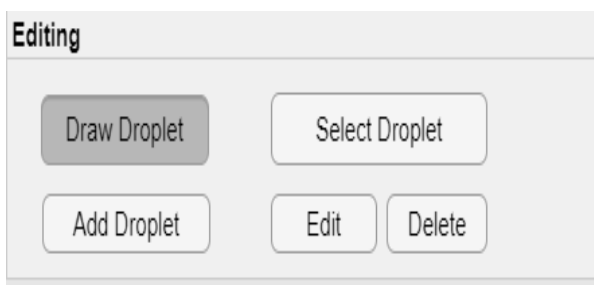
```

% Value changing function: ThresholdSlider
function ThresholdSliderValueChanging(app, event)
    changingValue = event.Value;
    app.ThresEditField.Value = changingValue;
    if app.ApplyThersholdButton.Value == 1
        if ~isempty(app.thimg)
            if app.IdentifyBoundaryButton.Value==1
                app.thimg = app.img > app.ThresEditField.Value;
                [app.centers,app.radii] = imfindcircles(app.thimg,[round(app.RminEditField.Value/app.Lpx)
round(app.RmaxEditField.Value/app.Lpx)],Sensitivity=app.SensEditField.Value,ObjectPolarity =
app.objp,EdgeThreshold=app.EdgeThresEditField.Value);    %,Method='twostage'
                imshow(app.thimg, 'Parent', app.UIAxes);
                showcircles(app)
                showmeasurements(app)
            else
                app.thimg = app.img > app.ThresEditField.Value;
                imshow(app.thimg, 'Parent', app.UIAxes);
            end
        else
            app.ApplyThersholdButton.Value = 0;
        end
    end
end
end

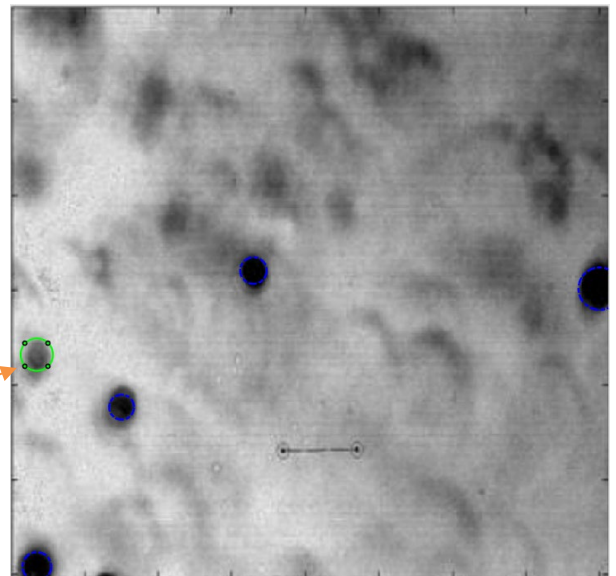
```

## VI. Adding or deleting a circle:

Since some droplets may be undetected, so we can manually add the droplets by drawing circles over them using the 'Draw Droplet' and 'Add Droplet' buttons. Similarly, we can delete and edit the detected droplets.

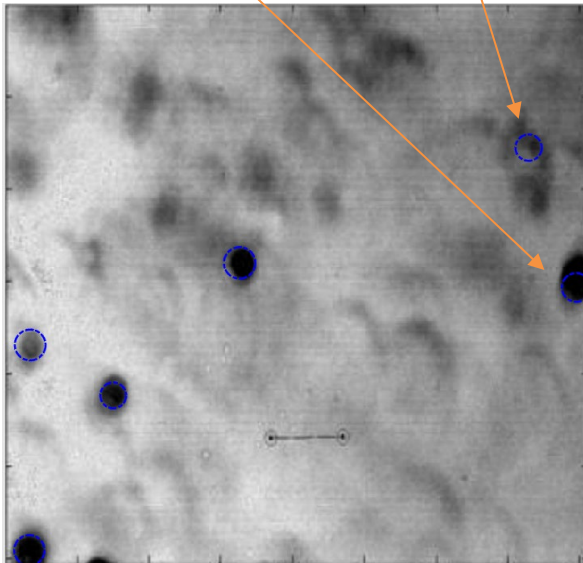


Adding a Droplet

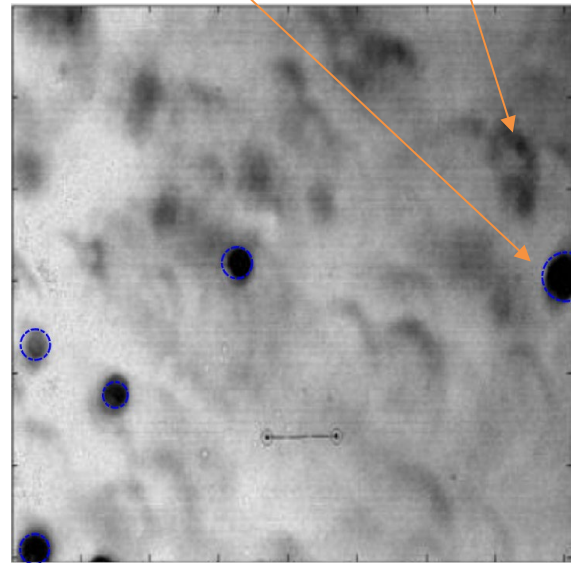


detected circle is of different size

Extra detected circle



The extra detected circle is deleted and circle of different size is edited using the edit and delete buttons.



```
% Value changed function: DrawDropletButton
function DrawDropletButtonValueChanged(app, event)
value = app.DrawDropletButton.Value;
if value == 1
if ~isempty(app.thimg)
delete(app.m);
app.SelectDropletButton.Value = 0;
app.h.Visible = 0;
showcircles(app);
app.a = drawcircle("Parent",app.UIAxes,'DrawingArea','unlimited','Color','g','FaceAlpha',0,'LineWidth',1.3,'MarkerSize',3);
else
app.DrawDropletButton.Value = 0;
end
elseif value == 0
delete(app.a);
end
end

% Button pushed function: AddDropletButton
function AddDropletButtonPushed(app, event)
if app.DrawDropletButton.Value == 1
app.h.Visible = 0;
radius = app.a.Radius;
center = app.a.Center;
if radius ~= 0
app.centers = [app.centers;center];
app.radii = [app.radii;radius];
end
end
```



```

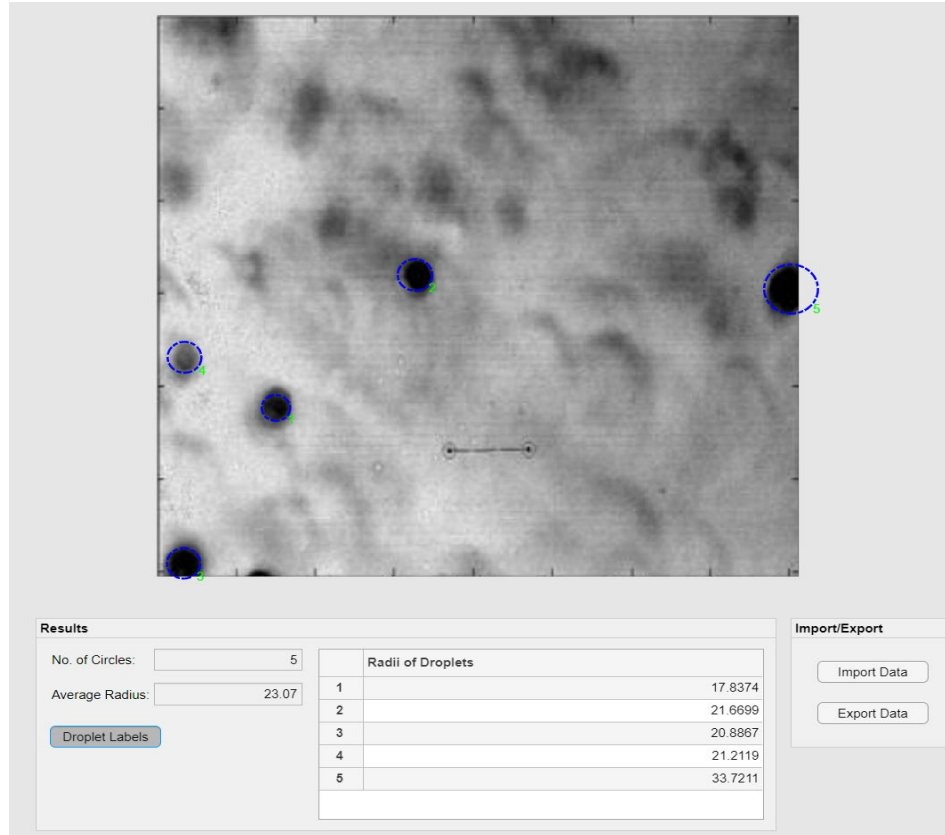
% Value changed function: SelectDropletButton
function SelectDropletButtonValueChanged(app, event)
if ~isempty(app.radii)
if app.DrawDropletButton.Value == 0
value = app.SelectDropletButton.Value;
if value == 1
app.cntr = app.centers;
app.rds = app.radii;
app.h.Visible = 0;
showcircles(app);
app.s = drawpoint("Parent",app.UIAxes);
x = app.s.Position(1);
y = app.s.Position(2);
delete(app.s);
n = length(app.radii);
xtemp = app.centers(1);
ytemp = app.centers(1+n);
temp = 1;
for i = 1:n
            if sqrt((x-xtemp)^2+(y-ytemp)^2) - sqrt((x-app.centers(i))^2+(y-app.centers(i+n))^2) >= 0
                xtemp = app.centers(i);
                ytemp = app.centers(i+n);
                temp = i;
            end
        end
        ctemp = [app.centers(temp),app.centers(temp+n)];
        rtemp = app.radii(temp);
        if sqrt((x-xtemp)^2+(y-ytemp)^2)<=rtemp
            app.cntr(temp,:) = [];
            app.rds(temp) = [];
            app.h.Visible = 0;
            app.h =
viscircles(app.UIAxes,app.cntr,app.rds,'EdgeColor','b','DrawBackgroundCircle',0,'LineStyle','-','LineWidth',1.5);
            close(gcf)
            app.m =
drawcircle("Parent",app.UIAxes,'DrawingArea','unlimited','Color',[0.66,0.81,1.00],'LineWidth',1.3,'FaceAlpha',0,'MarkerSize'
,3,"Center",ctemp,Radius=rtemp);
        else
            app.SelectDropletButton.Value = 0;
        end

        elseif value == 0
            delete(app.s);
            delete(app.m);
            app.h.Visible = 0;
            showcircles(app);
            showmeasurements(app);
        end
        elseif app.DrawDropletButton.Value == 1
            app.SelectDropletButton.Value = 0;
        end
        elseif isempty(app.radii)
            app.SelectDropletButton.Value = 0;
        end
end
end

```

## VII. Results:

The Results section of the application shows the 'Average Radius' of the detected droplets and also shows radius of individual droplet with label.



```
function showmeasurements(app)
    app.UITable.Data = app.Lpx*app.radii;
    app.NoofCirclesEditField.Value = length(app.radii);
    if ~isempty(app.radii)
        app.AverageRadiusEditField.Value = mean(app.Lpx*app.radii);
    else
        app.AverageRadiusEditField.Value = 0;
    end
    if app.DropletLabelsButton.Value == 1
        n = length(app.radii);
        Label = string(1:n);
        x = app.centers(1:n);
        y = app.centers(n+1:2*n);
        dx = app.radii(1:n);
        dx = (4/5)*(dx. ');
        delete(app.txt);
        app.txt = text(x+dx,y+dx,Label,'Color','g','Parent',app.UIAxes);
    end
end
```

## VIII. Import and Export data:

The data can be imported and exported with the help of 'import' and 'export' buttons. The exported data can be used by other devices with this app installed.

```
% Button pushed function: ImportDataButton
function ImportDataButtonPushed(app, event)
    [imp_file,imp_path] = uigetfile('*.mat','Import Data');
    if isequal(imp_file,0)
        return
    end
    b = load(fullfile(imp_path,imp_file));
    cla(app.UIAxes,"reset");
    app.centers = b.a1;
    app.radii = b.a2;
    app.img = b.a3;
    app.ThresEditField.Value = b.a4;
    app.SensEditField.Value = b.a5;
    app.EdgeThresEditField.Value = b.a6;
    app.RminEditField.Value = b.a7;
    app.RmaxEditField.Value = b.a8;
    app.FileNameEditField.Value = b.a9;
    app.lpx = b.a10;
    app.objp = b.a11;
    if size(app.objp,2) == 4
        app.DropDown.Value = "Black Circles";
    else
        app.DropDown.Value = "White Circles";
    end
    app.file = app.FileNameEditField.Value;
    app.ThresholdSlider.Value = app.ThresEditField.Value;
    app.SensitivitySlider.Value = app.SensEditField.Value;
    app.EdgeThresholdSlider.Value = app.EdgeThresEditField.Value;
    app.thimg = app.img;
    app.LengthpxEditField.Value = app.lpx;
    app.LengthEditField.Value = 0;
    app.PixelsEditField.Value = 0;
    app.ApplyThersholdButton.Value = 0;
    if isempty(app.radii)
        app.IdentifyBoundaryButton.Value = 0;
    else
        app.IdentifyBoundaryButton.Value = 1;
    end
    app.DropletLabelsButton.Value = 0;
    app.DrawDropletButton.Value = 0;
    app.SelectDropletButton.Value = 0;
    app.DrawLineButton.Value = 0;
```

```

        imshow(app.thimg, 'parent', app.UIAxes);
        showcircles(app)
        showmeasurements(app)
    end

    % Button pushed function: ExportDataButton
    function ExportDataButtonPushed(app, event)
        if isequal(app.file,0)
            return
        end
        [~,newfile] = fileparts(app.file);
        fname = sprintf("%s_DataSet.mat",newfile);
        [filename,export_path] = uiputfile(fname,"Export Data");
        if isequal(export_path,0)
            return
        end
    end
    a1 = app.centers;
    a2 = app.radii;
    a3 = app.img;
    a4 = app.ThresEditField.Value;
    a5 = app.SensEditField.Value;
    a6 = app.EdgeThresEditField.Value;
    a7 = app.RminEditField.Value;
    a8 = app.RmaxEditField.Value;
    a9 = app.FileNameEditField.Value;
    a10 = app.Lpx;
    a11 = app.objp;

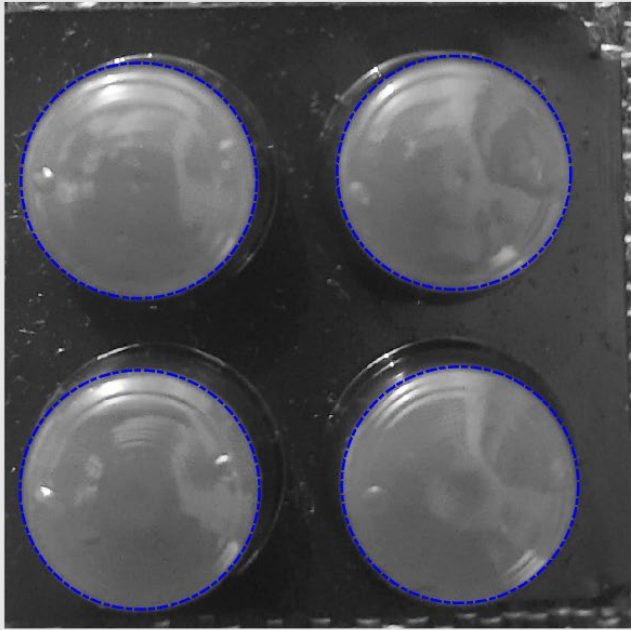
    save(fullfile(export_path,filename), 'a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'a10', 'a11')
end

```

## Validation:

For validating the average radius, we tested the application on images with known average radius.

Known Radius	Calculated Radius	Accuracy	Error
0.65cm	0.6594 cm	98.55%	1.45%



**Load**

Browse
Reset

File Name : testing1(3.5\_3.5\_1.2cm).jpg

**Calibration**

Length: 3.5

Pixels: 1280

Length/px: 0.002734

Plot Calibration Line

Calibrate

**Range**

Draw Line

Find Length

Line Length: 236.2

Rmin: 200 Rmax: 250

**Results**

No. of Circles: 4

Average Radius: 0.6594

Droplet Labels

Radii of Droplets	
1	0.6520
2	0.6576
3	0.6680
4	0.6601

**Import/Export**

Import Data

Export Data

**Controls**

Apply Thresh

Refined Image

Thres: 79.39

Identify Boundary

Sensitivity

White Circles

EdgeThreshold

Sens: 0.9783 EdgeThres: 0.3525

**Editing**

Draw Droplet

Select Droplet

Add Droplet

Edit

Delete

8. **Concluding Remarks:** The application is completed and can be downloaded from [here](#).

9. References:

- <https://journals.sagepub.com/doi/full/10.1177/1756827716640244>
- <https://in.mathworks.com/help/images/ref/imfindcircles.html>

10. Declaration: I declare that no part of this report is copied from other sources. All the references are properly cited in this report.

Rahman

Signature of the Student

Signature of the Supervisor

### **Supervisor's Recommendation for the Evaluation**

Please tick any one of the following

1. The work done is satisfactory, and sufficient time has been spent by the student. The submission by the student should be evaluated in this term.
2. The work is not complete. Continuity Grade should be given to the student. The student would need to be evaluated in the next semester for the same Design Project with me.
3. The work is not satisfactory. There is no need for evaluation. The students should look for another Design Credit Project for the next semester.
4. [Other Comment, if 1-3 are not valid] \_\_\_\_\_

**Signature of the Supervisor**