**Dataset**

Extracted data from dataset – 1 can be found from file “data\_set1.xlsx” generated from first hand experiments. Extracted data from dataset – 2 (LINK) can be found from the file “data\_set2.xlsx”.

**Data Extraction**

A custom program was made using Visual Basic, to extract pixel values of the different layers of graphene. Clicking a point on the image gives the RGB values of the particular pixel. Different pixel values of each image were stored in a text file.

|  |  |
| --- | --- |
|  |  |

*Figure 1 – Data extraction software*

The text file was then converted into a csv file.

**Data Extraction**

Two datasets were utilized in the current study. Dataset – 1 contained 18 labeled images of graphene on SiO2 substrate for a particular lighting condition and magnification level. The scripts were developed focusing on Dataset – 1 and then projected into Dataset2 which contained 110 images of graphene on SiO2 with different conditions of lighting and magnification levels. The annotations for Dataset – 2 were contained in a JSON file. The decryption of the JSON file was done via a MATLAB script. Following this step, the JSON data was loaded and parsed. The segmentation coordinates of all the images were thereafter grabbed. Polygons were drawn according to these coordinates and specific colors were assigned to the regions of various categories. The categories were ‘monolayer’, ‘few layers’, and ‘many layers’ of graphite flakes. The ‘monolayer’ was assigned a color of yellow, ‘few layers’ was assigned green, and ‘many layers’ were assigned red. Images with newly assigned polygons were saved. A custom visual basic program was used to extract pixel value of individual graphene layers from each image of both the datasets. Digital color images are made of pixels and pixels are made of combinations of primary colors generally represented by RGB. Among the R, G and B values, the G values showed a prominent pattern. Fig – 1 is a plot for Dataset – 1. The green channel is subjected to less noise compared to red and blue channel. As an image is made from bright to dark, the G value decreases consistently compared to the R and G values. Brightness primarily depends on the G values.

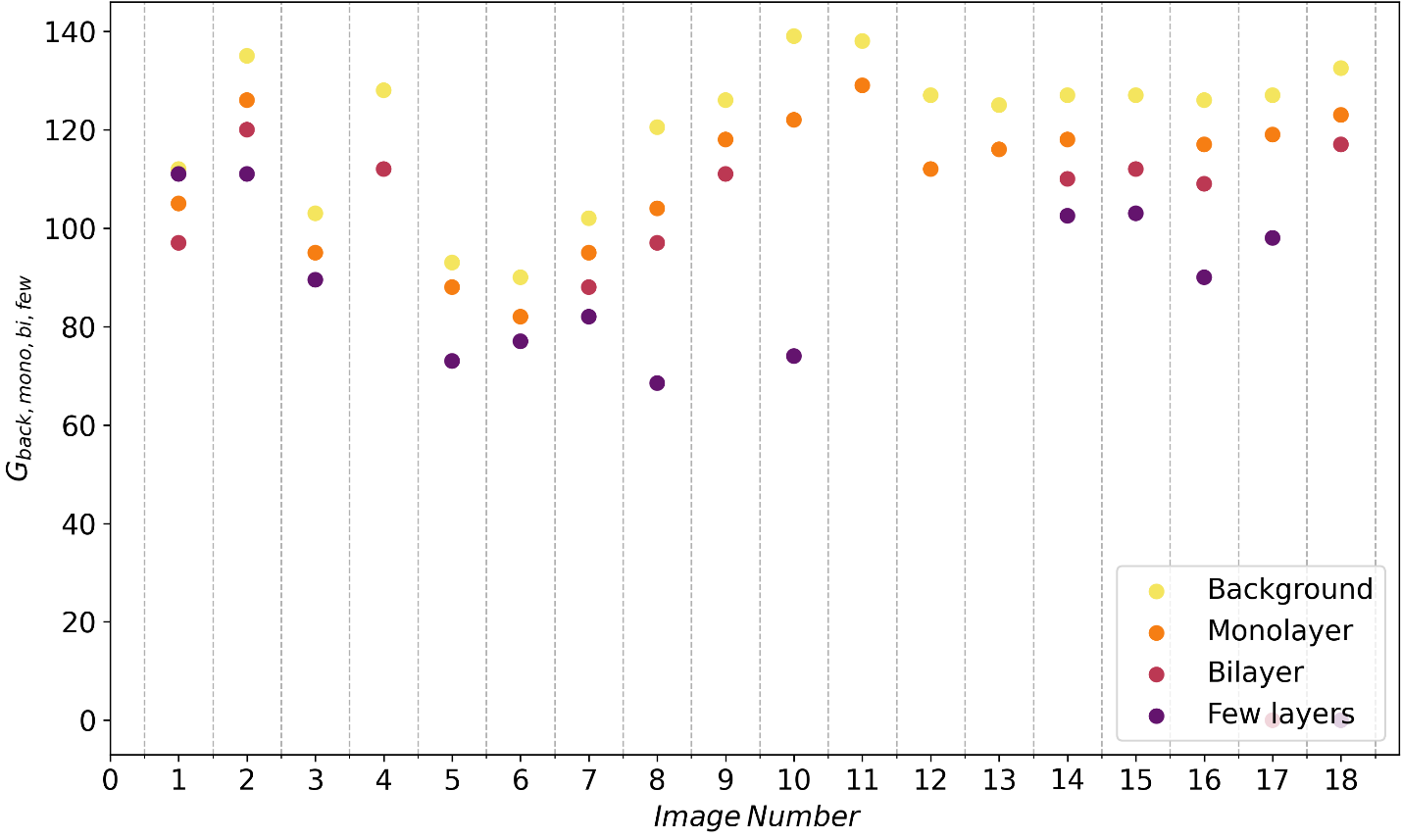


Figure 1: Scatter plot showing similar order of G values of various layers of each image.

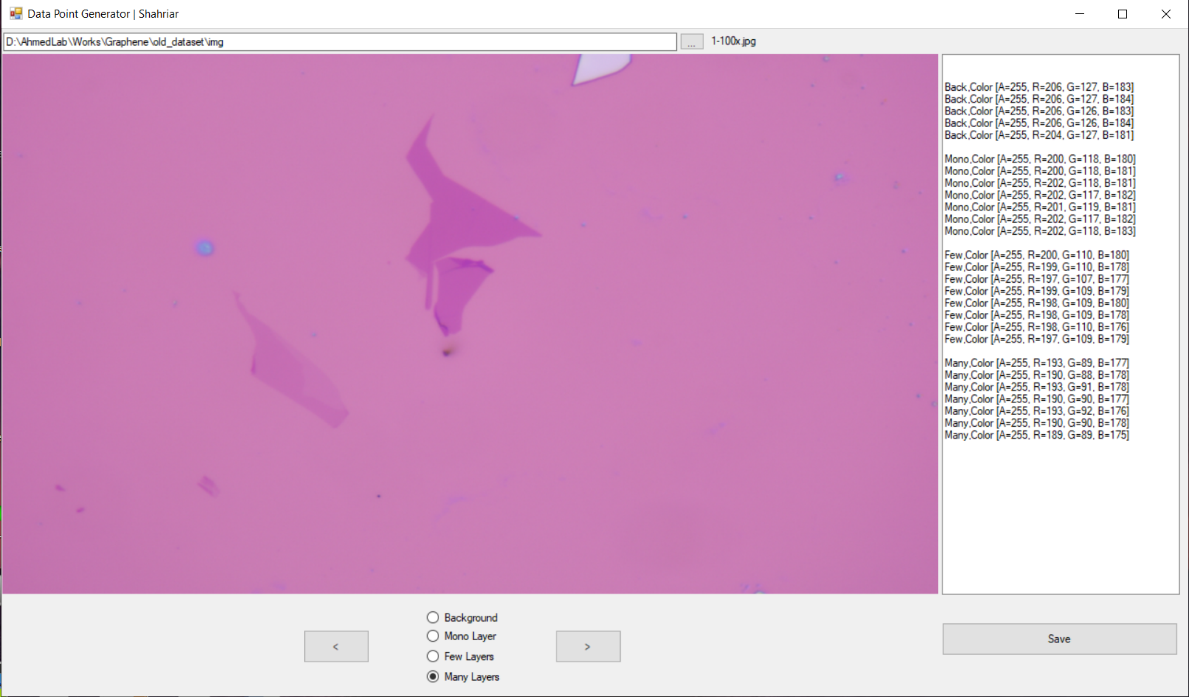
****

Figure 2: Pixel value extraction using a home-grown software

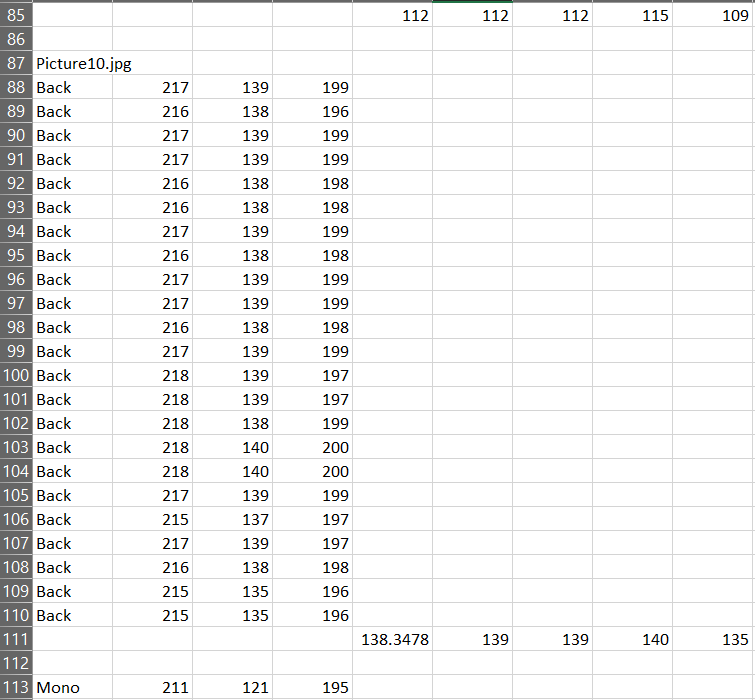


Figure 2. Data saved to CSV file

**Normality Test**

From the csv file, only the G values of the pixels were taken. Then taking only the median of the G values, a new excel file was created. A normality check was done and the data were found to be

normally distributed.

Figure 3. Normality check

**Analysis**   
The G values of the various layers were determined. The ratio of G values of mono, few or many layers with the background were taken. The minimum, maximum and standard deviation of these ratios determined.

|  |  |  |
| --- | --- | --- |
| Mono/Back | Bi/Back | Few/Back |
| 0.9375 | 0.866071 | X |
| 0.9333333 | 0.888889 | 0.822222 |
| 0.9223301 | X | 0.868932 |
| X | 0.875 | X |
| 0.9462366 | X | 0.784946 |
| 0.9111111 | X | 0.855556 |
| 0.9313725 | 0.862745 | 0.803922 |
| 0.8630705 | 0.804979 | 0.568465 |
| 0.9365079 | 0.880952 | X |
| 0.8776978 | X | 0.532374 |
| 0.9347826 | X | X |
| 0.8818898 | X | X |
| 0.928 | X | X |
| 0.9291339 | 0.866142 | 0.807087 |
| X | 0.88189 | 0.811024 |
| 0.9285714 | 0.865079 | 0.714286 |
| 0.9370079 | X | 0.771654 |
| 0.9283019 | 0.883019 | X |

|  |  |  |  |
| --- | --- | --- | --- |
| max= | 0.9462366 | 0.888889 | 0.868932 |
| min= | 0.8630705 | 0.804979 | 0.532374 |
|  |  |  |  |
| mean= | 0.920428 | 0.867477 | 0.758224 |
| median= | 0.9288526 | 0.870571 | 0.803922 |
| STDV= | 0.0236328 | 0.022547 | 0.105714 |
| Lrange | 0.8731623 | 0.822382 | 0.546795 |
| Urange | 0.9676936 | 0.912571 | 0.969653 |
| variance= | 0.0005585 | 0.000508 | 0.011176 |

From this data, the upper and lower thresholds were calculated.

**Identification**

A script using MATLAB was developed which at first identifies the G value of the background using histogram. Then the script reads each pixel values and calculates the ratio of G value of that pixel with the background. This ratio is then compared with the values determined from the excel sheet. From this comparison, each pixel will be assigned a distinct color which will enable us to differentiate the various layers from one another. This is the segmented image.

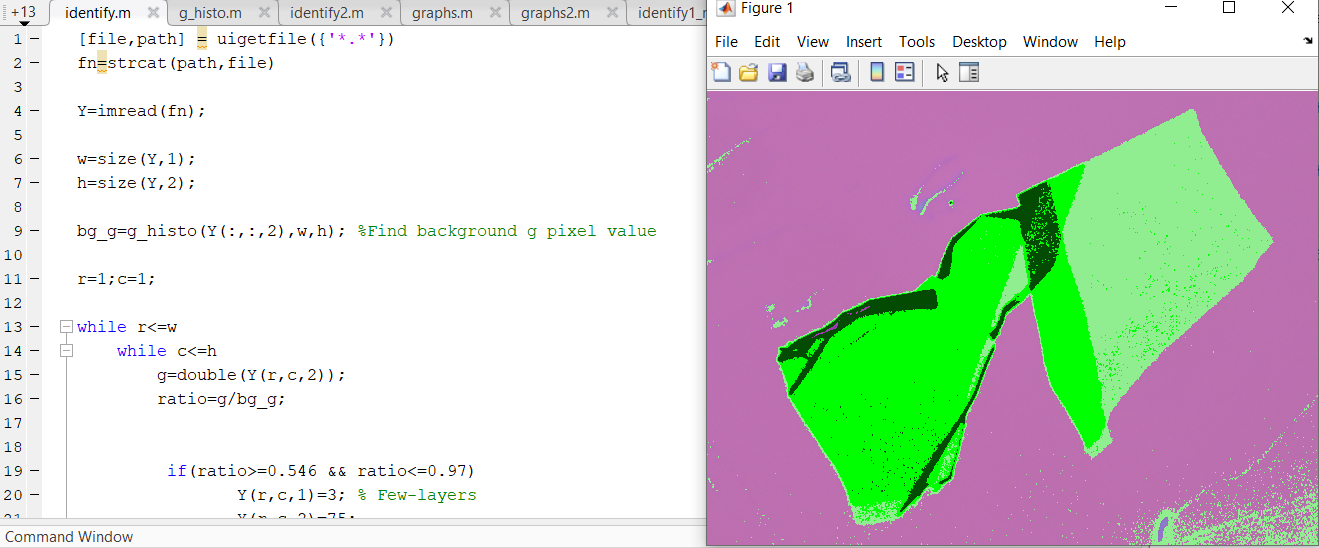


Figure 4. MATLAB Script and Segmented Image

MATLAB script for layer identification

|  |
| --- |
| [file,path] = uigetfile({'\*.\*'})  fn=strcat(path,file)    Y=imread(fn);    w=size(Y,1);  h=size(Y,2);    bg\_g=g\_histo(Y(:,:,2),w,h); %Find background g pixel value    r=1;c=1;    while r<=w  while c<=h  g=double(Y(r,c,2));  ratio=g/bg\_g;        if(ratio>=0.327 && ratio<=0.49)  Y(r,c,1)=3; % Few-layers  Y(r,c,2)=75;  Y(r,c,3)=3;  end    if(ratio>=0.489 && ratio<=0.897)  Y(r,c,1)=0; % Bilayer  Y(r,c,2)=255;  Y(r,c,3)=0;  end      if(ratio>=0.777 && ratio<=0.953)  Y(r,c,1)=144; % Monolayer  Y(r,c,2)=238;  Y(r,c,3)=144;  end          %if(ratio>=0.388 && ratio<=0.530)  % Y(r,c,1)=255; % Multi ^ Bi  % Y(r,c,2)=0;  % Y(r,c,3)=0;  %end        c=c+1;  end  r=r+1;  c=1;  end    imshow(Y) |

MATLAB script for accuracy determination

|  |
| --- |
| % Load the image using a prompt  [filename, pathname] = uigetfile('\*.\*', 'Select an Image File');  img = imread(fullfile(pathname, filename));  %img = imread('D:\AhmedLab\Works\Graphene\old\_dataset\outputs\old\_linear\recolour\2.png');    % Reshape the input image into a 2D matrix  img\_2d = reshape(img, [], 3);      %Count number of mono,bi and few pixels    m\_color = [246, 126, 19];  b\_color = [188, 56, 63];  f\_color = [100, 20, 110];    m\_mask = ismember(img\_2d, m\_color, 'rows');  b\_mask = ismember(img\_2d, b\_color, 'rows');  f\_mask = ismember(img\_2d, f\_color, 'rows');    m\_px = nnz(m\_mask);  b\_px = nnz(b\_mask);  f\_px = nnz(f\_mask);    disp(['Total num of monolayer pixels [', num2str(m\_color), ']: ', num2str(m\_px)]);  disp(['Total num of bilayer pixels [', num2str(b\_color), ']: ', num2str(b\_px)]);  disp(['Total num of fewlayers pixels [', num2str(f\_color), ']: ', num2str(f\_px)]);  %Counting done    mono\_out=m\_px;  bi\_out=b\_px;  few\_out=f\_px;      imshow(img);  % Allow user to draw a polygon on the image  while true  % Prompt user for layer input  layer\_input = input('Which layer? (1/2/3): ', 's');    % Terminate loop if input is not 1, 2, or 3  if ~(layer\_input == '1' || layer\_input == '2' || layer\_input == '3')  break  end    % Convert layer input to numeric value  layer = str2double(layer\_input);    % Allow user to draw a polygon on the image  h = impoly();  mask = createMask(h);      % Extract the masked region of interest for each color channel  masked\_red\_channel = img(:,:,1) .\* uint8(mask);  masked\_green\_channel = img(:,:,2) .\* uint8(mask);  masked\_blue\_channel = img(:,:,3) .\* uint8(mask);    % Concatenate the masked R, G, and B values into a single matrix  masked\_rgb\_values = cat(2, masked\_red\_channel(:), masked\_green\_channel(:), masked\_blue\_channel(:));    % Check which rows have an RGB value of (255, 0, 0)  rgb\_target\_mono = [246,126,19];  rgb\_target\_bi = [188,56,63];  rgb\_target\_few = [100,20,110];  is\_target\_mono = ismember(masked\_rgb\_values, rgb\_target\_mono, 'rows');  is\_target\_bi = ismember(masked\_rgb\_values, rgb\_target\_bi, 'rows');  is\_target\_few = ismember(masked\_rgb\_values, rgb\_target\_few, 'rows');    % Count the number of pixels with the target RGB value  num\_target\_pixels\_mono = sum(is\_target\_mono);  num\_target\_pixels\_bi = sum(is\_target\_bi);  num\_target\_pixels\_few = sum(is\_target\_few);    if(layer\_input=='1')  mono\_out=mono\_out-num\_target\_pixels\_mono;  elseif(layer\_input=='2')  bi\_out=bi\_out-num\_target\_pixels\_bi;  elseif(layer\_input=='3')  few\_out=few\_out-num\_target\_pixels\_few;  end      total\_px\_inside=nnz(mask);  disp(['Mono pixels in this zone: ', num2str(num\_target\_pixels\_mono)]);  disp(['Bi pixels in this zone: ', num2str(num\_target\_pixels\_bi)]);  disp(['Few pixels in this zone: ', num2str(num\_target\_pixels\_few)]);    accuracy\_mono=num\_target\_pixels\_mono\*100/total\_px\_inside;  accuracy\_bi=num\_target\_pixels\_bi\*100/total\_px\_inside;  accuracy\_few=num\_target\_pixels\_few\*100/total\_px\_inside;    disp(['Mono: ',num2str(accuracy\_mono)]);  disp(['Bi: ',num2str(accuracy\_bi)]);  disp(['Few: ',num2str(accuracy\_few)]);      clear is\_target\_mono is\_target\_bi is\_target\_few;      end    ac\_mono=(m\_px-mono\_out)\*100/m\_px;  ac\_bi=(b\_px-bi\_out)\*100/b\_px;  ac\_few=(f\_px-few\_out)\*100/f\_px;    % save accuracy values to a text file  filename\_acr = 'accuracy.txt';  fid = fopen(filename\_acr, 'a');  fprintf(fid, '%s\n', filename); % save image filename  fprintf(fid, 'Mono: %f\n', ac\_mono); % save mono accuracy  fprintf(fid, 'Bi: %f\n', ac\_bi); % save bi accuracy  fprintf(fid, 'Few: %f\n', ac\_few); % save few accuracy  fclose(fid); |