Supporting Information

Low-Cost and High-Speed Fabrication of Camouflage-Enabling Microfluidic Devices using Ultra High Molecular Weight Polyethylene

Xiaoruo Sun[1], Asad Asad[1]*, Mehnab Ali[1], Luka Morita[1], Patricia I. Dolez[2], James D. Hogan[1], Dan Sameoto[1]*

[1] Department of Mechanical Engineering, University of Alberta, Edmonton, Canada

[2] Department of Human Ecology, University of Alberta, Edmonton, Canada

* Corresponding authors: Dan Sameoto (sameoto@ualberta.ca) & Asad Asad (aaasad@uaberta.ca)

Keywords: Microfluidic devices, Adaptive visible camouflage, Adaptive Infrared camouflage, Polyethylene, Flexible microfluidic devices.

Table S1. Summarizes the cutting settings of the Silhouette Cameo 4 craft cutter software for different channel configurations. These settings include blade number, force, speed, and pass. The blade determines the depth of the blade's cut, ranging from 1-10, with 10 being the deepest. The force setting controls the amount of pressure the blade applies vertically, while the speed setting controls how quickly the machine operates. The pass setting represents the number of times the machine cuts the pattern.

	W1, W2, and W3 Channel	H2 and H3 Channel	Channel with Metal layer
Blade Number	3	9	3
Force	20	30	20
Speed	4	4	4
Passes	1	1	1

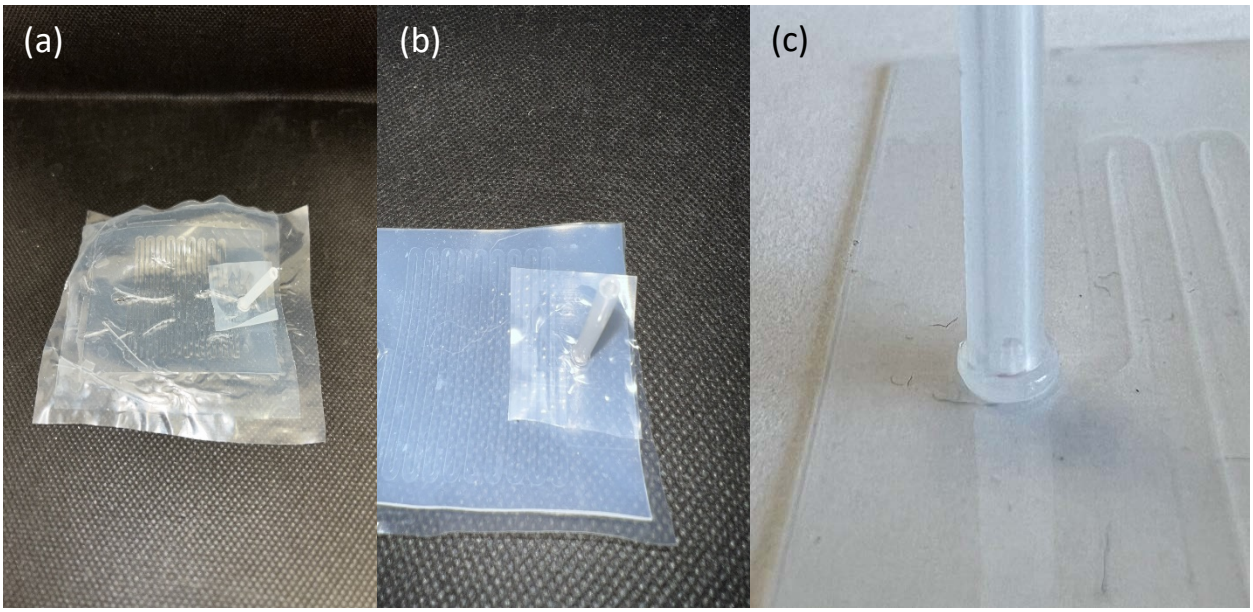

Figure S1. (a) early attempts at bonding UHMWPE without polyester resulted in buckling and wrinkling of the sheet, which often led to damage or blockage of the microfluidic channel. Furthermore, the early macro-to-micro interface tubing technique depicted in (a) and (b) was complex and unreliable, frequently resulting in permanent sealing of the inlet and outlet or significant leakage. However, significant progress has been made in the design of the interface, as demonstrated in (c). The current design features a robust PE tube thermally bonded to the UHMWPE microfluidic device with no leakage, thanks to the use of a polyester sheet that distributes force and eliminates buckling and wrinkling during the bonding process. This significant improvement in the design of the microfluidic device underscores the importance of continued optimization and refinement of fabrication processes in order to achieve reliable and durable microfluidic devices for a wide range of applications.

Figure S2. Shows a microfluidic device microfluidic device with an incorporated metalized sheet

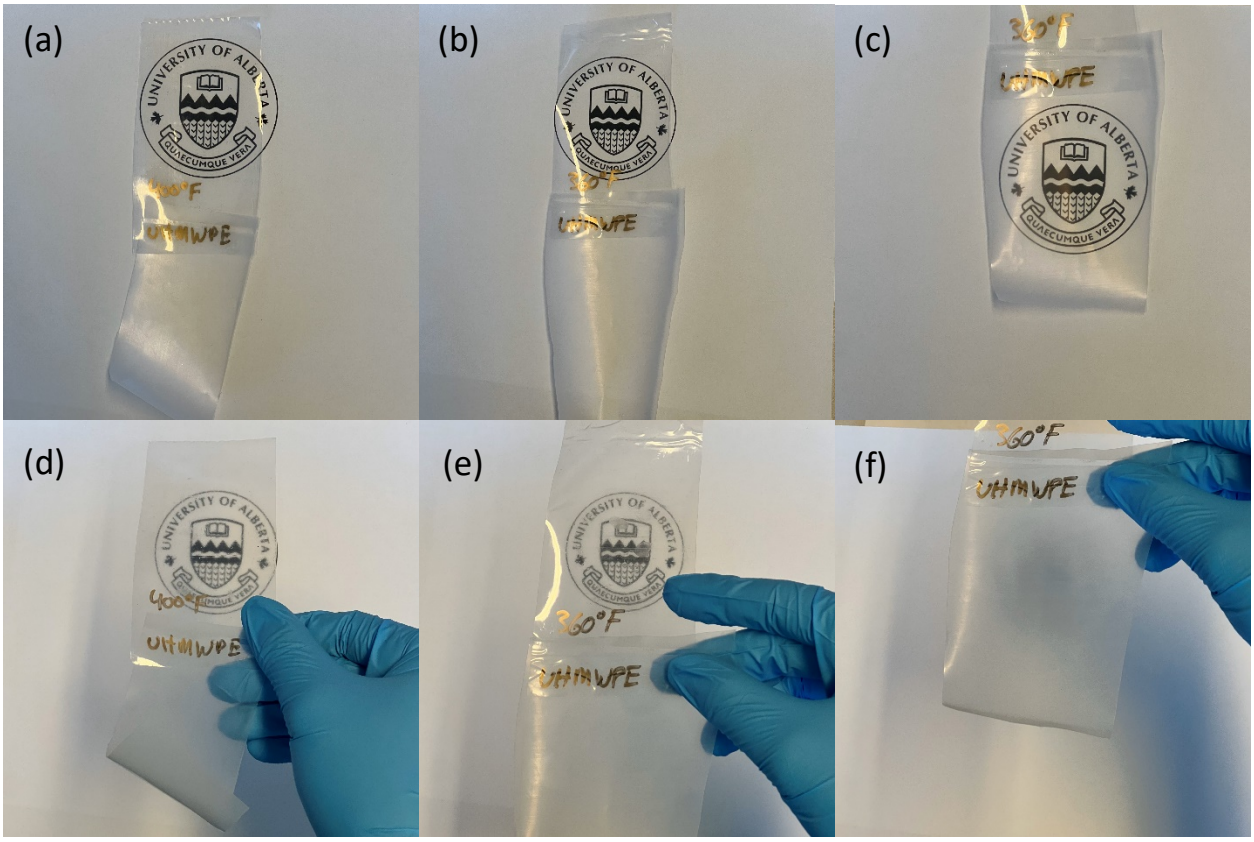

Figure S3. demonstrates the results of different surface treatment temperatures by heat-pressing a 100 μm UHMWPE sheet on a smooth surface such as Kapton tape or polyester sheet. The treatment reduces surface roughness and eliminates internal gaps in the UHMWPE. The treated and untreated UHMWPE sheets are then bonded using an impulse sealer, with the upper piece being the treated sheet and the lower piece being the original untreated sheet. The specimens were labeled with temperatures of 360°F and 400°F. The results show that there is no significant difference in transparency between the treated sheets at these two temperatures. However, there is a significant difference in transparency between the treated and untreated sheets. When placed directly on the University of Alberta logo, both the treated and untreated UHMWPE sheets allow the logo to be seen through them (as demonstrated in (a), (b), and (c)). However, when placed roughly 100 mm vertically away from the logo, the treated UHMWPE sheets still allow the logo to be visible (as shown in (d) and (e)), while the untreated UHMWPE sheet becomes opaque and the logo is no longer visible (as shown in (f)).

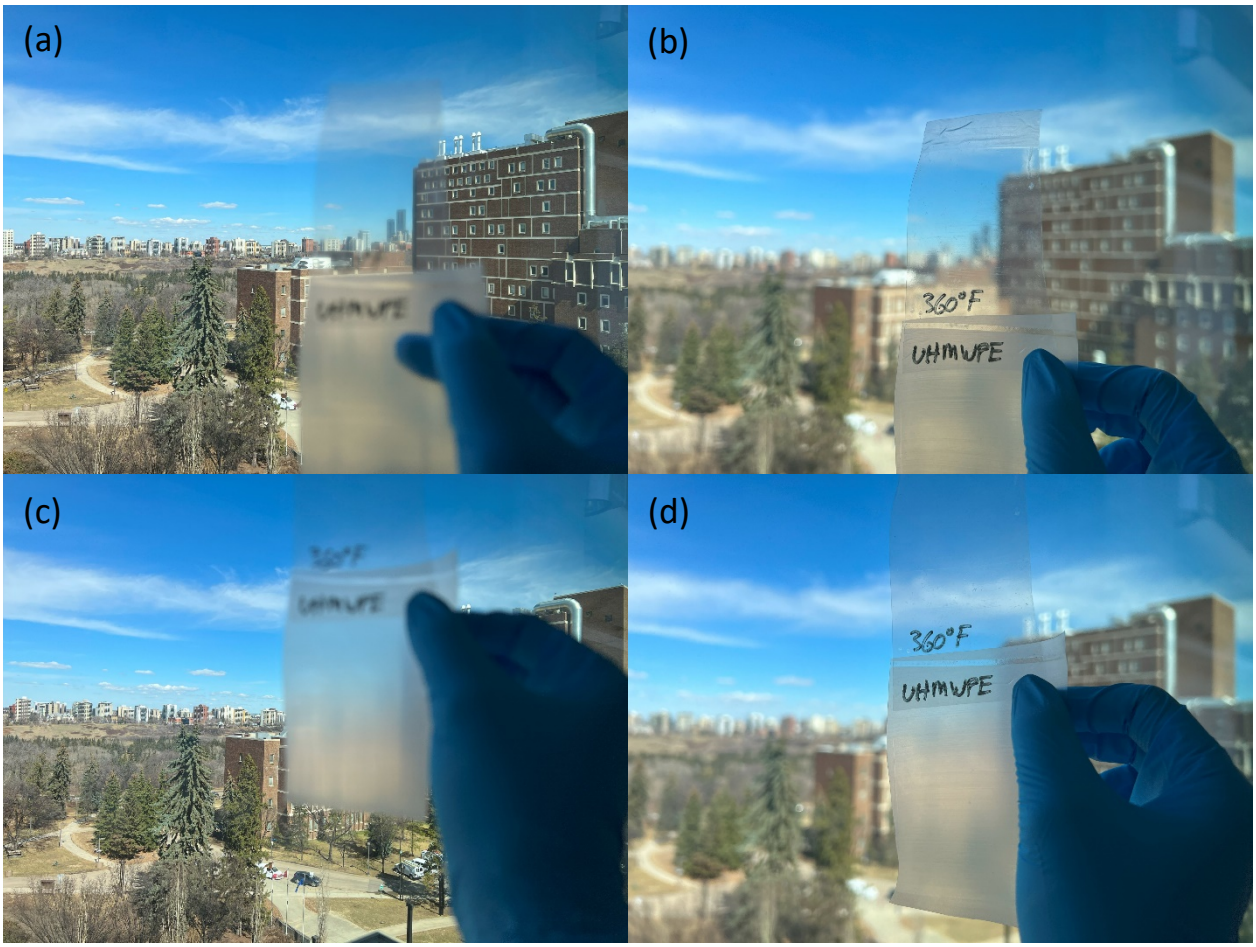

Figure S4. demonstrates the transparency comparison between treated and untreated UHMWPE sheets, confirming that the treated sheet has higher transparency. Panels (a) and (c) show the far building through the treated sheet, while panels (b) and (d) focus on the sheet itself. The untreated sheet is opaque, while the scenes outside the University of Alberta window are visible only through the treated UHMWPE sheet.

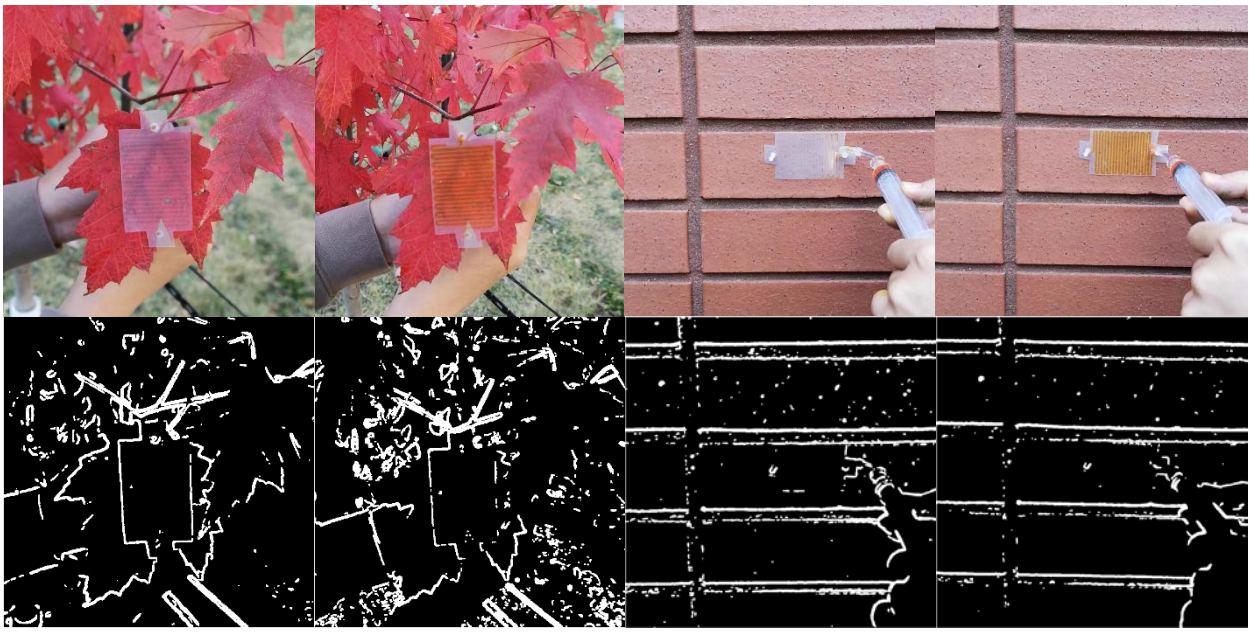

Figure S5. Shows the Canny edge-finding algorithm analysis results for the brick and leaf backgrounds. The results for these two backgrounds were not significant since the canny edge finding algorithm was only suitable for some certain environments. With these two backgrounds, only the major edges were detected, the microfluidic device demonstrated no major improvements in terms of edge hiding. However, this result still demonstrated the microfluidic device cannot be detect by edge finding algorithm since the filled and unfilled device has very similar results, there is no detected edges within both the filled and unfilled device areas.

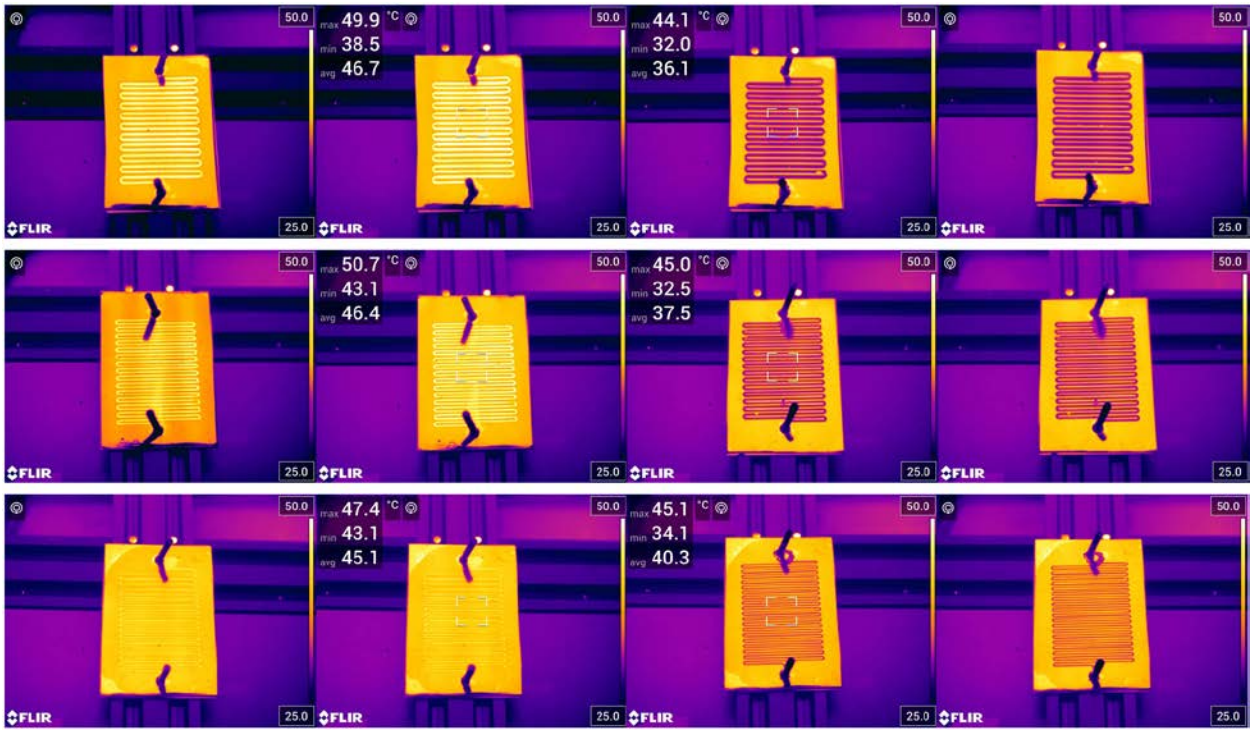

Figure S6. Shows the raw images directly captured from the FLIR IR camera, the first row is the W3, the second row is W2, and the third row is W1 configuration. The first column is the unfilled channels, the second and third columns are the images with the measured data, and the fourth column is the filled channel IR appearance. The temperature bar is configured as 25-50℃ to present better contrast.

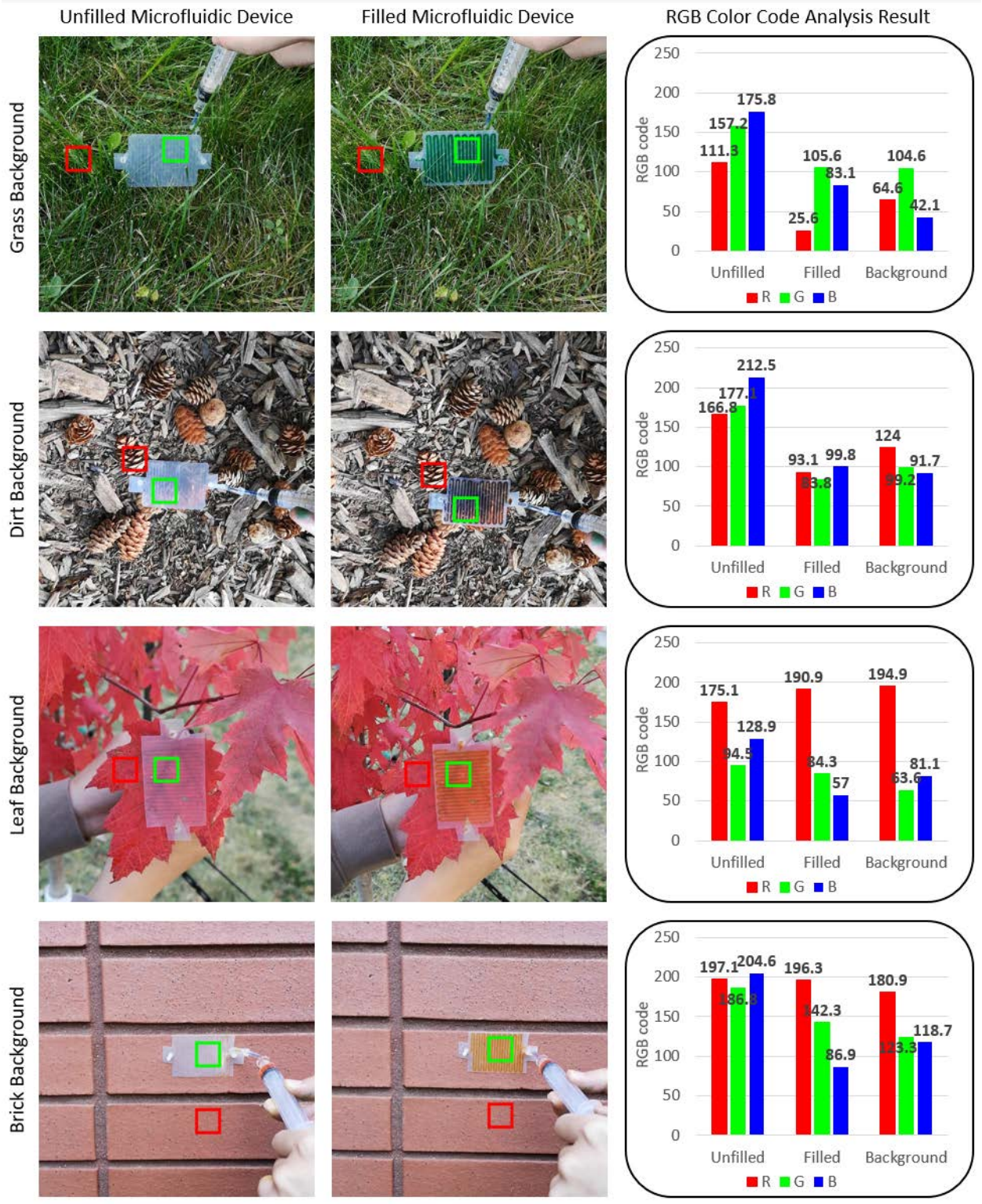

Figure S7. Shows the detailed RGB value in the third column.

Code for canny edge finding:

```
% Read an image and convert it to grayscale
img = imread('green0.png');
gray = rgb2gray(img);

% Apply a Gaussian filter to the image
gauss = imgaussfilt(gray, 3);

% Compute the gradient magnitude and direction using Sobel operators
[Gx, Gy] = imgradientxy(gauss);

% Compute the gradient magnitude and direction
gradient = sqrt(Gx.^2 + Gy.^2);
direction = atan2(Gy, Gx);

% Non-maximum suppression
suppressed = nlfilter(gradient, [3 3], @(x) max(x(:)));

% Double thresholding
high_threshold = 0.25 * max(suppressed(:));
low_threshold = 0.05 * high_threshold;
potential_edges = zeros(size(gradient));
potential_edges(suppressed > high_threshold) = 1;

% Edge tracking by hysteresis
final_edges = zeros(size(gradient));
while any(potential_edges(:))
    [r, c] = find(potential_edges, 1);
    final_edges(r, c) = 1;
    potential_edges(r, c) = 0;
    for rr = -1:1
        for cc = -1:1
            if r+rr > 0 && r+rr <= size(gradient, 1) && c+cc > 0 && c+cc <=
size(gradient, 2) && final_edges(r+rr, c+cc) ~= 1 && suppressed(r+rr, c+cc) >
low_threshold
                final_edges(r+rr, c+cc) = 1;
                potential_edges(r+rr, c+cc) = 0;
            end
        end
    end
end

% Show the edges
figure;
imshow(final_edges);
```

Code for color matching:

```
clc;
clear;
% Read in the image file
img = imread('red0.png');

% Define the size and location of the two regions
size_region = [100,100];
location1 = [580,330];
location2 = [580,510];

% Crop the regions from the image
region1 = img(location1(1):location1(1)+size_region(1)-1,
location1(2):location1(2)+size_region(2)-1, :);
region2 = img(location2(1):location2(1)+size_region(1)-1,
location2(2):location2(2)+size_region(2)-1, :);

% Calculate the mean color difference between the two regions
%diffR = mean((mean(region1(:,:,1)) - mean(region2(:,:,1)))*10/255);
%diffG = mean((mean(region1(:,:,2)) - mean(region2(:,:,2)))*10/255);
%diffB = mean((mean(region1(:,:,3)) - mean(region2(:,:,3)))*10/255);
%diff = (abs(diffR) + abs(diffG) + abs(diffB))/3;
% Plot the image
imshow(img);
hold on;

% Plot the first region
rectangle('Position', [location1(2), location1(1), size_region(2), size_region(1)],
'EdgeColor', 'r','LineWidth', 10);
%text(location1(2)+size_region(2)/2, location1(1)+size_region(1)/2, 'Region 1',
'Color',
'white','HorizontalAlignment','center','VerticalAlignment','middle','fontsize',14,'fo
ntweight','bold');

% Plot the second region
rectangle('Position', [location2(2), location2(1), size_region(2), size_region(1)],
'EdgeColor', 'g','LineWidth', 10);
%text(location2(2)+size_region(2)/2, location2(1)+size_region(1)/2, 'Region 2',
'Color',
'white','HorizontalAlignment','center','VerticalAlignment','middle','fontsize',14,'fo
ntweight','bold');

% Print the color difference value
region1R = mean((mean(region1(:,:,1))));
region1G = mean((mean(region1(:,:,2))));
region1B = mean((mean(region1(:,:,3))));
region2R = mean((mean(region2(:,:,1))));
region2G = mean((mean(region2(:,:,2))));
region2B = mean((mean(region2(:,:,3))));
disp(region1R);
disp(region1G);
disp(region1B);
disp(region2R);
disp(region2G);
disp(region2B);
```