ACCSCIENCE PUBLISHING

International Journal of Bioprinting

RESEARCH ARTICLE

A pilot evaluation of a three-dimensional bioprinted tumor model for preclinical assessment of electroporation-based therapies

Supplementary file

Matlab scripts

1. Analysis of porosity

```
%% 1. Load the image
% Replace 'YourImageFile.jpeg' with the path to your image file
img = imread('YourImageFile.jpeg');
%% 2. Convert to grayscale if necessary
if size(img, 3) == 3
    img_gray = rgb2gray(img);
else
    img_gray = img;
end
%% 3. Interactive scale calibration (10 μm)
disp('Draw a line over the 10 μm scale bar, then press the button to continue.');
fig = figure;
imshow(img_gray);
title('Draw a line on the 10 μm scale bar');
h = imdistline(gca);  % Create interactive line
uicontrol('Style', 'pushbutton', 'String', 'Finish Calibration', ...
    'Position', [20 20 140 30], ...
    'Callback', 'uiresume(gcbf)');
uiwait(gcf);  % Wait for user interaction
pixels_per_10um = getDistance(h);  % Get the length of the drawn line in pixels
close(fig);  % Close the figure window
% Calculate pixel-to-micrometer conversion factor
um_per_pixel = 10 / pixels_per_10um;
disp(['Measured length: ', num2str(pixels_per_10um, '%.2f'), ' pixels per 10 μm']);
disp(['Conversion factor: ', num2str(um_per_pixel, '%.4f'), ' μm/pixel']);
```

```
%% 4. Crop the image to exclude the scale bar (adjust if necessary)
[rows, cols] = size(img_gray);
crop_height = round(0.88 * rows);  % Adjust cropping factor as needed
img_cropped = img_gray(1:crop_height, :);

figure; imshow(img_cropped, []); title('Cropped image (scale bar excluded)');
%% 5. Apply Gaussian filter to reduce noise
sigma = 1;
img_filtered = imgaussfilt(img_cropped, sigma);
figure; imshow(img_filtered, []); title('Filtered image');
%% 6. Binarize the image using Otsu's method with a scaling factor
threshold = graythresh(img_filtered) * 0.70;  % Adjust factor based on image contrast
img_binary = imbinarize(img_filtered, threshold);
%% 7. Invert the binary image so pores are white
img_binary = ~img_binary;
%% 8. Display the binary map
figure;
imagesc(img_binary);
axis equal tight;
title('Binary map (0: material, 1: pores)');
colorbar;
try
    colormap([0 0 0; 1 1 1]);
    caxis([0 1]);
catch
    warning('Colormap not applied.');
end
%% 9. Count pixels
total_pixels = numel(img_binary);
pores_pixels = sum(img_binary(:));
material_pixels = total_pixels - pores_pixels;
disp(['Total pixels: ', num2str(total_pixels)]);
disp(['White pixels (pores): ', num2str(pores_pixels)]);
disp(['Black pixels (material): ', num2str(material_pixels)]);
%% 10. Calculate porosity percentage
porosity = (pores_pixels / total_pixels) * 100;
disp(['Porosity: ', num2str(porosity, '%.2f'), '%']);
%% 11. Compute average pore area excluding small artifacts
min_area = 50;  % Minimum area in pixels squared
img_binary_clean = bwareaopen(img_binary, min_area);
labeled_img = bwlabel(img_binary_clean);
stats = regionprops(labeled_img, 'Area');
pore_areas_pixel = [stats.Area];
average_area_pixel = mean(pore_areas_pixel);
%% 12. Convert average pore area to μm²
average_area_um2 = average_area_pixel * (um_per_pixel^2);
disp(['Average pore area (pixels²): ', num2str(average_area_pixel, '%.2f')]);
disp(['Average pore area (μm²): ', num2str(average_area_um2, '%.2f')]);
```

2. Fluorescence quantification of programmed death-ligand 1 signal intensity

```
% === GENERAL PARAMETERS ===
% Define the folder containing your image data
folder = 'your_folder_path_here';  % <-- Replace this with your path
% === DEFINE GROUPS AND FILES ===
% Create a structure where each field represents a group
% and contains a cell array of filenames (relative to the folder)
groups = struct();
% Example (commented out — users should fill in their own groups):
% groups.GroupA = { 'image1.png', 'image2.png' };
% groups.GroupB = { 'image3.png', 'image4.png' };
% === IMAGE INTENSITY ANALYSIS ===
group_names = fieldnames(groups);
mean_values = zeros(1, numel(group_names));
std_values = zeros(1, numel(group_names));
```

```
for g = 1:numel(group_names)
    group = group_names{g};
    files = groups.(group);
    intensities = zeros(1, numel(files));

    for i = 1:numel(files)
        file_path = fullfile(folder, files{i});
        if exist(file_path, 'file')
            img = imread(file_path);
            if size(img, 3) == 3
                img = rgb2gray(img);
            end
            if size(img,1) > 155
                img = img(1:end-155, :);  % Optional crop
            end
            intensities(i) = mean(img(:));
        else
            warning('File not found: %s', file_path);
            intensities(i) = NaN;
        end
    end
    valid = ~isnan(intensities);
    if any(valid)
        mean_values(g) = mean(intensities(valid));
        std_values(g) = std(intensities(valid));
    else
        mean_values(g) = NaN;
        std_values(g) = NaN;
    end
end
% % === PLOT RESULTS ===
% figure;
% b = bar(mean_values);
% hold on;
% b.FaceColor = 'flat';
%
% % Generate default colors
% colors = lines(max(1, numel(group_names)));
% b.CData = colors(1:numel(group_names), :);
% % Add error bars
% errorbar(1:numel(group_names), mean_values, std_values, ...
%     'k.', 'LineWidth', 1.5);
% % Customize axes
% xticks(1:numel(group_names));
%
% if isempty(group_names)
%     xticklabels("Example Group 1");  % Placeholder if no groups defined
% else
%     xticklabels(strrep(group_names, '_', ' '));
% end
% xtickangle(30);
% xlabel('Name of Groups');
% ylabel('Mean Intensity Rate (a.u.)');
% title('Image Intensity Quantification Across Groups');
% grid on;
```