```
function diff_mat = find_bg_color(input_im, N)

  % This function returns a matrix of weights with the portion of region that is
likely to be background.
  % This is accomplished by first reducing the image to 8 colors (2 colors per
channel).
  % Then, we find the most frequent color within each region, and measure the
portion of pixels that contain that color.
  % The concept being that if the most frequent color within a region dominates that
region, then that region is likely to be background.

  im_size = size(input_im);
  num_rows = im_size(1);
  num_cols = im_size(2);
  row_length = floor(num_rows/N);
  col_length = floor(num_cols/N);

  num_pixels = row_length*col_length;
  diff_mat = zeros(N,N);

  for i = 0 : N - 1

    for j = 0 : N - 1

      temp_im = input_im(1 + i*row_length : (i+1)*row_length, 1 + j*col_length :
(j+1)*col_length, 1 : 3);
      temp = uint8(floor(temp_im/128)*255);


      rgb_columns = reshape(temp, [], 3); %the following four lines of code count
the number of instances of the most frequent color
      [unique_colors, m, n] = unique(rgb_columns, 'rows'); %this code is taken from
the mathworks website as posted by Steve Eddins
      color_counts = accumarray(n, 1);  %the original article is available here:
      [max_count, idx] = max(color_counts);
%https://blogs.mathworks.com/steve/2008/01/31/counting-occurrences-of-image-colors/


      diff_mat(i+1,j+1) = 1 - max_count/num_pixels;

    endfor

  endfor

endfunction
```