```
function [region_matrix N S_mat prob_mat] = identify_fine_features(I)

%This is another version of identify_features that uses a different initial image
partition function that identifies smaller features

[N Y X] = maximize_std_dev(I) %used to calculate N, where N^2 is the number of
regions the image is partitioned into

[A B C D] = score_image(I,N,X,Y); %generates a series of vectors, though we use only
D, which is a measure of information contained in each region

temp_mat = vec2mat(D,N); %turns D into a matrix where each entry corresponds to the
information content of region (i,j) of the image partition

S_mat = generate_S_mat_distribution(temp_mat); %generates a score from 0 to 1 of how
notable the information content of each region is

s_smat = std(S_mat(:)); %the std dev of S_mat, which is used as the basis for
determining how similar two regions are

bg_mat = find_bg_color(I,N); %a matrix (bg_mat) that contains a measure of how
dominated each region is by its most frequent color

prob_mat = generate_S_mat_distribution(bg_mat); %generates a score from 0 to 1 of
how dominated each region is by its most frequent color using bg_mat

delta_bg = 1; %a dummy variable reserved for future use

s_bg = std(prob_mat(:)); %also reserved for future use

symm_1 = left_right_delta(prob_mat(:)'); %a measure of the prominance of the
background

symm_2 = left_right_delta(S_mat(:)'); %a measure of the prominance of regions with
unusual levels of information content

symm = inf_weighted_prob([symm_1 symm_2]);

num_iterations = 25; %a cap on how many times the algorithm will test the test of
the entropy of the assemblage

divisor = max(25, N^(N*(1-symm))/10); %the divisor of the standard deviation of the
S_mat matrix

increment = (s_smat - (s_smat/divisor))/num_iterations;

i = 0;
```

```
delta_max = (s_smat/divisor) + i*increment;
region_matrix_1 = gather_regions_delta(N,S_mat,prob_mat,delta_max,i*delta_bg);
%assembles the image into groups of regions

delta_max = (s_smat/divisor) + (i+1)*increment;
region_matrix_2 = gather_regions_delta(N,S_mat,prob_mat,delta_max,(i+1)*delta_bg);

temp = (entropy(region_matrix_1) - entropy(region_matrix_2))^2/increment; %an
approximation to the derivative of the entropy of the assemblage as a function of i

delta = temp;

region_matrix = region_matrix_2;

H_max = log2(N^2);
H_min = .47*H_max + .53*((divisor - 25)/divisor)*H_max;

while((entropy(region_matrix_1) >= H_min) && (i <= num_iterations)) %the more
diverse the image, the more information we'll want in the region_matrix, generating
smaller features

  i = i + 1;

  region_matrix_1 = region_matrix_2;

  delta_max = (s_smat/divisor) + (i+1)*increment;
  region_matrix_2 = gather_regions_delta(N,S_mat,prob_mat,delta_max,(i+1)*delta_bg);

  delta = (entropy(region_matrix_1) - entropy(region_matrix_2))^2/increment;

  if(delta > temp)

    temp = delta;
    region_matrix = region_matrix_2;

  endif

endwhile

endfunction
```