```
function s = test_entropy_array_3D(input_array, N)
%takes an array of 3-space vectors as inputs, and returns the standard deviation of
the information contents of the categories
%resulting from a partition into regions of length by max-min/N

 [x num_items] = size(input_array);
 [X Y Z] = generate_data_space(input_array);

  %we add a bit of cushion on either side to ensure that there are no errors due to
precision

  min_x = X(1) - .0001;
  min_y = Y(1) - .0001;
  min_z = Z(1) - .0001;

  max_x = X(2) + .0001;
  max_y = Y(2) + .0001;
  max_z = Z(2) + .0001;

  delta_x = (max_x - min_x) / N;
  delta_y = (max_y - min_y) / N;
  delta_z = (max_z - min_z) / N;

  count_matrix = zeros(N,N,N);

%account for the minimum elements of each dimension, just use a seperate test within
the loop for i,j,k, == 1
%iterates through the entire array and tests to which category each element belongs

 for m = 1 : num_items

   temp_vector = input_array{m};

   for i = 1 : N

     for j = 1 : N

       for k = 1 : N

         in_category_x = 0;
         in_category_y = 0;
         in_category_z = 0;

         cat_min_x = min_x + (i-1)*delta_x;
         cat_max_x = min_x + i*delta_x;

         cat_min_y = min_y + (j-1)*delta_y;
         cat_max_y = min_y + j*delta_y;
```

```
 cat_min_z = min_z + (k-1)*delta_z;
 cat_max_z = min_z + k*delta_z;

if(i == 1)

    if(cat_min_x <= temp_vector(1) && temp_vector(1) <= cat_max_x)

      in_category_x = 1;

    endif

 elseif(cat_min_x < temp_vector(1) && temp_vector(1) <= cat_max_x)

    in_category_x = 1;

 endif

 if(j == 1)

    if(cat_min_y <= temp_vector(2) && temp_vector(2) <= cat_max_y)

      in_category_y = 1;

    endif

 elseif(cat_min_y < temp_vector(2) && temp_vector(2) <= cat_max_y)

    in_category_y = 1;

 endif

 if(k == 1)

    if(cat_min_z <= temp_vector(3) && temp_vector(3) <= cat_max_z)

      in_category_z = 1;

    endif

 elseif(cat_min_z < temp_vector(3) && temp_vector(3) <= cat_max_z)

    in_category_z = 1;

 endif

 if(in_category_x && in_category_y && in_category_z)
```

```
            count_matrix(i,j,k) = count_matrix(i,j,k) + 1;

          endif

        endfor

      endfor

    endfor

 endfor

 %this portion of the code calculates the standard deviation of the information
contents of the partition

 count_vector = count_matrix(:);

 num_categories = N^3;

 for i = 1 : num_categories

   if(count_vector(i) > 0)

      count_vector(i) = spec_log(num_items / count_vector(i));

   else

      count_vector(i) = 0;

   endif

 endfor

 s = std(count_vector);

endfunction
```