```
function [ y ] = kmeans_color( x,K )
%COLOR_INDEX Puts the pixels of x into k clusters and outputs as y
%   x - RGB image vector of any size
%   k - number of clusters
%
%   Uses the iterative k-means method to quantize an image to a limited
%   number of colors

MAX_IT = 10; %max number of iterations
ep = 10; %max error allowed in solution
iteration = 0; %iterations counter
errorsum = 11;

[M,N,c] = size(x);
labels = zeros(M,N);
centroids = uint8(zeros(1,K,3));
centroids(1,:,1) = uint8(linspace(0,256,K));
centroids(1,:,2) = uint8(linspace(0,256,K));
centroids(1,:,3) = uint8(linspace(0,256,K));

y = zeros(M,N,c); %copy image into y
x = double(x); %convert x to doubles
%centroids = double(centroids);

sums = zeros(1,K,3);
counts = zeros(1,K);



while (iteration <= MAX_IT && errorsum > ep)

    errorsum = 0;

    for m = 1:M
        for n = 1:N      %for each pixel

            smallest_dist = 500;
            closestk = 1;

            for k = 1:K   %check distance from each centroid

                %distance formula
                dist = sqrt((x(m,n,1) - double(centroids(1,k,1)))^2 + (x(m,n,2) - double(cent
roids(1,k,2)))^2 + (x(m,n,3) - double(centroids(1,k,3)))^2);

                if dist < smallest_dist

                  smallest_dist = dist; %smallest_dist is the current smallest distance to a
centroid for this pixel
                  closestk = k;


                end %end if
```

```
            end %centroid loop



            %%%updates
            errorsum = errorsum + smallest_dist; %update total error
            labels(m,n) = closestk;

            counts(closestk) = counts(closestk) + 1; %update new label count
            sums(1,closestk,1) = sums(1,closestk,1) + x(m,n,1);
            sums(1,closestk,2) = sums(1,closestk,2) + x(m,n,2);
            sums(1,closestk,3) = sums(1,closestk,3) + x(m,n,3);

            y(m,n,1) = centroids(1,closestk,1);
            y(m,n,2) = centroids(1,closestk,2);
            y(m,n,3) = centroids(1,closestk,3);


        end %column

    end %row

    for k = 1:K
      if counts(k) > 0
        centroids(1,k,1) = sums(1,k,1) / counts(k);
        centroids(1,k,2) = sums(1,k,2) / counts(k);
        centroids(1,k,3) = sums(1,k,3) / counts(k); %update centroid values
      else
         centroids(1,k,1) = centroids(1,k,1) / 10;
         centroids(1,k,2) = centroids(1,k,2) / 2;
         centroids(1,k,3) = centroids(1,k,3) / 3;     %randomly reassign
      end

    end %Change location of centroid k

    iteration = iteration + 1
    errorsum = errorsum / (M*N)


end %one iteration of the k-mean process


imshow(uint8(y))


end
```

Published with MATLAB® R2015b