```
%============================== segKmeans ==============================
%
%  function [J, K, means] = segKmeans(I, numk, iter, means)
%
%  Perform k-means segmentation on the grayscale image I.
%
%  Input:
%    I          - image I ( from |R^2 -> |R^d ).
%    numk    - The number k to use in k-means.
%    iter       - Maximum number of iterations (can stop earlier if no change).
%    means      - initial guess at means, each column is a mean value.
%              means should be provided row-wise.  [k x d matrix]
%    ncov       - covariance matrix to use in the scaling (use 1 if none).
%                      can be scalar for all means to use, or can be a unique
%                      value for each mean value.
%
%  Output:
%    J          - the segmentation map.
%    K          - the simplified image using the means and the segmentation.
%    means      - the final segmentation means.
%
%============================== segKmeans ==============================

%
%  Name:                segKmeans.m
%
%  Author:              Patricio A. Vela, pvela@gatech.edu
%
%  Created:             2010/01/05
%  Modified:    2012/04/07
%
%============================== segKmeans ==============================
function [J, K, means] = kmeans(I, iter, numk, means)

%--[0] Parse the input arguments, set to defaults if needed.
if (nargout == 0)               % If nothing expected, then don't bother
  return;                               %   doing the computations.
end

if (nargin < 3)                 % If first three not given, can't do much.
  disp('ERROR: Need at least the first three arguments');
  error('BadArgs');
end


%--[1] Prep workspace and variables.  Convert image data to vector data.
sz = size(I);
layers = 1
if length(sz) == 3
    layers = sz(3);
end
vec = reshape(I, sz(1) * sz(2), layers);
%vec = I(:);
J = zeros(sz);
%--[2] Perform k-means clustering.

if ( (nargin > 3) && ( numk == size(means,1) ) )
  % kmeans invocation goes here.  Make sure to set the optional
  % arguments properly for both iterations and initial guess.
```

```
  % Make sure to grab all that's needed for the triple output.
  [J, means] = kmeans(vec, numk, 'MaxIter', iter, 'Start', means);
else
  % let Matlab guess means.  Make sure to set the optional
  % arguments properly for iterations.
  % Make sure to grab all that's needed for the triple output.
  [J, means] = kmeans(vec, numk, 'MaxIter', iter);
end

J = reshape(J, sz(1), sz(2), 1);

%--[3] Prep additional output.  Convert cluster indices into actual
%      image data values by using the returned means.  This will
%      generate an image using only the k mean values.


if ((nargin == 3) || isempty(ncov))
  ncov = 1;                     % If no covariance, default is 1.
end

if (isscalar(ncov))             % If scalar, copy for each mean value.
  ncov = repmat(ncov, [layers, numk])';
end

%means = means .* ncov;
if (nargin > 1)
  K = J;
  for i=1:numk
    K(J == i) = means(i,1);
  end
end

end
```

---

*Published with MATLAB® R2016b*