```
%================================ segBayes ================================
%
%  function [J, K, means] = segBayes(I, iter, means, stdvs)
%
%  Perform Bayesian segmentation on the image I.
%
%  Input:
%    I          - image I.
%    iter       - the number of iterations. if negative, then go to convergence.
%    means      - initial guess at means (column-wise).
%    stdvs      - optional input of standard deviations.
%             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 Gaussian means for each class.
%    stdvs      - the Gaussian standard deviations for each class.
%    iter       - number of iterations (if < 0, then to convergence).
%
%================================ segBayes ================================
function [J, K, means] = segBayes(I, iter, means, stdvs)

%--[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 < 4)         % If first three not given, can't do much.
  disp('ERROR: Need at least the first three arguments');
  error('BadArgs');
end

if (isempty(iter))                              % If iterations not given,
  iter = -1;                                    %  then go to convergence.
end

if ((nargin < 4) || isempty(stdvs))             % If no argument or if empty.
  stdvs = 1;
end

if (isscalar(stdvs))                                    % If scalar, vectorize.
  stdvs = repmat(stdvs, [1, size(means, 2)]);
end

%--[1] Prep workspace and variables.
[M, N] = size(I);               % Get problem dimensions.
P = length(means);

                                        % Homogeneous priors.
priors = reshape(ones([M*N,P])*diag(ones(P,1)/P), [M, N, P]);
image  = I(:,:,ones(P,1));              % Duplicate image I for quick evaluation.

                                                        % Init. space for (posterior) probabilities.
post   = reshape(ones([M*N,P])*diag(ones(P,1)/P), [M, N, P]);

%--[2] Setup, then perform the segmentation iterations.
hasConverged = false;
oJ = zeros([M, N]);
mold = means;

while ( ((iter < 0) && (~hasConverged)) || ( iter > 0) )
```

```
  % YOUR CODE HERE.
  % Steps:
  %  (1) Calculate likelihoods using normal distribution.
  post = compLikelihoods();

  %  (2) Multiply by priors to generate posterior "probs".
  post = post .* priors;

  %  (3) Normalize posteriors.
  normalizeProb();

  %  (4) Smooth posteriors (use heat/diffusion or Gaussian smoothing).
  post = smoothProb(post);

  %  (5) Normalize smoothed posteriors.
  normalizeProb();

  %  (6) Find maximum posterior for classification/segmentation.
  [val, J] = max(post,[],3);

  %  (7) Update means.
  compMeans();

  %  (8) Set priors to be posteriors.
  priors = post;

  %  (8) Termination conditions and loop update.
  if all(mold==means)                          % All new means equal to old means.
    hasConverged = true;
  elseif all(oJ(:) == J(:))                    % Old segmentation equals new segmentation.
    hasConverged = true;
  end

  if (iter > 0)
    iter = iter - 1;
  end

end

ncov = stdvs .^ 2;
if (nargout >= 2)
  K = J;
  means = means .* ncov;
  for i=1:size(means,2)
      K(J == i) = means(i);
  end
end

%
%============================ Helper Functions ============================
%
%  These functions live within the scope of the segBayes function.
%  What that means is that they can be invoked from the loop above
%  and they will have access to the variables above (think of them
%  as global variables in some sense).  Using functions within a
%  function is a clean way to do complex thing but have the main
%  code of the function look nice and clean.
%

  %-------------------------- compLikelihoods --------------------------
  %
  %  Computes the likelihoods for Bayesian inference.
  %
  function likelies = compLikelihoods

  likelies = zeros([M, N, P]);          % Initialize.
```

```
  for li = 1:P
    likelies(:,:,li) = normpdf(I,repmat(means(li),M,N),repmat(stdvs(li),M,N)); %TODO
  end

  end
  %---------------------------- smoothProb ----------------------------
  %
  %  Smooth the probabilities as part of the optimization step in Bayesian
  %  segmentation.  You can invoke whatever function you would like here.
  %  If you have seprate code to do that, just insert the function call here
  %  with all of the parameters set properly.
  %
  %  This function can either smooth each individual probability field (for
  %  each of the classes), or smooth all of them at once.  It is up to you
  %  to decide that part and code things appropriately above.
  %
  function iprob = smoothProb(iprob)

  % Invoke your favorite smoother here to smooth the probabilities.
  sigma = 0.5;
  iprob = imgaussfilt(iprob, sigma);

  end

  %--------------------------- normalizeProbs ---------------------------
  %
  %  Normalize the posterior probabilities.
  %
  function normalizeProb

  tmp  = sum(post,3) + eps;     % Compute sum.  Add epsilon to avoid divide
                                %  by zero.
  tmp  = tmp(:,:,ones(P,1));
  post = post./tmp;             % Divide by sum (plus epsilon).

  end

  %---------------------------- compMeans ----------------------------
  %
  %  Given the segmentation, compute the class means.  The mean should
  %  be updated only if there is more than 1 pixel in the class.
  %
  function compMeans
  for mi=1:length(means)                                % For each class label ...
    pts_in_layer = sum(sum(J == mi));
    if (pts_in_layer > 0)
        means(mi) = sum(I(J == mi)) ./ pts_in_layer;
    else
        means(mi) = 0;
    end
  end
  end

end

%
%================================ segBayes ================================
```

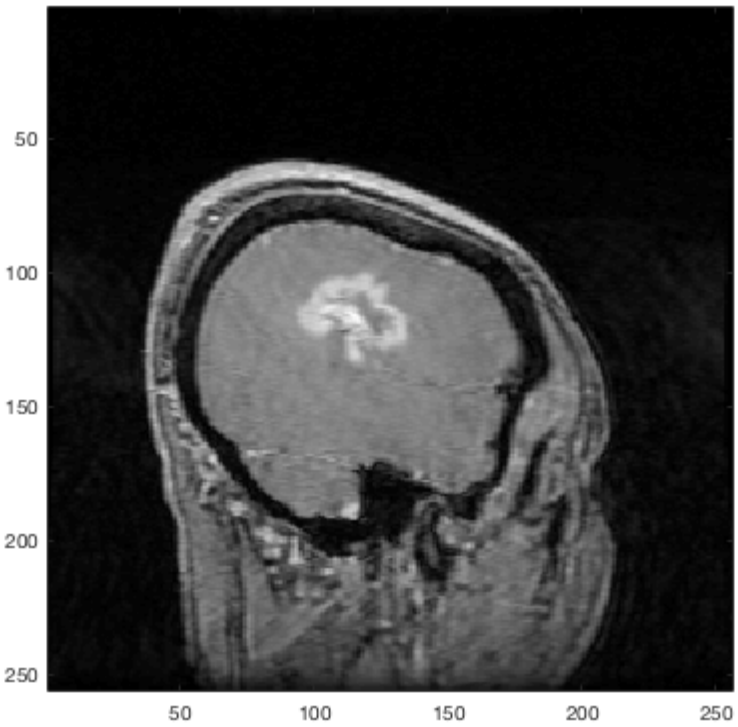
Published with MATLAB® R2016a

```
%============================== segmentB ==============================
%
%  script segmentB
%
%
%  This is a Matlab script that will run the Bayesian relaxation
%  for segmenting an image.   You should select two images from the
%  homework Matlab file and perform segmentation on them.
%
%  This code uses some Matlab tricks to be somewhat generic.  First,
%  all arguments are encapsulated into a cell array.  This works as
%  follows.  The cell array belows consists of two arguments:
%
%  >> sampleCellArray = {40, 34};
%
%  that when expanded as an argument to a function, provides two
%  inputs to the function,
%
%  >> plus( sampleCellArray{:} )
%
%  The output should be the addition of the two arguments:
%
%  ans =
%
%     74
%
%  Anyhow, this function expectes the same, but the arguments are
%  consistent with what the segKmeans function expects.
%
%============================== segmentB ==============================
close all
load('segment.mat');
picks = [1 2];

for i = 1:length(picks)
  switch (picks(i))
    case 1,
      images{i} = westin;
      iparms{i} = { 10 , [3, 10, 60, 140, 150], 1 };
    case 2,
      images{i} = fish04;
      iparms{i} = { 10 , [1, 5 ,40, 80, 120], 1 };
  end
end


for i = 1:length(images)
  [segimg{i}, K, nmeans{i}] = segBayes(double(images{i}), iparms{i}{:});

  figure(3*i-2);
  imagesc(images{i});
  colormap('gray');
  axis('image');

  figure(3*i-1);
  imagesc(segimg{i});
  colormap('default');
  axis('image');
```

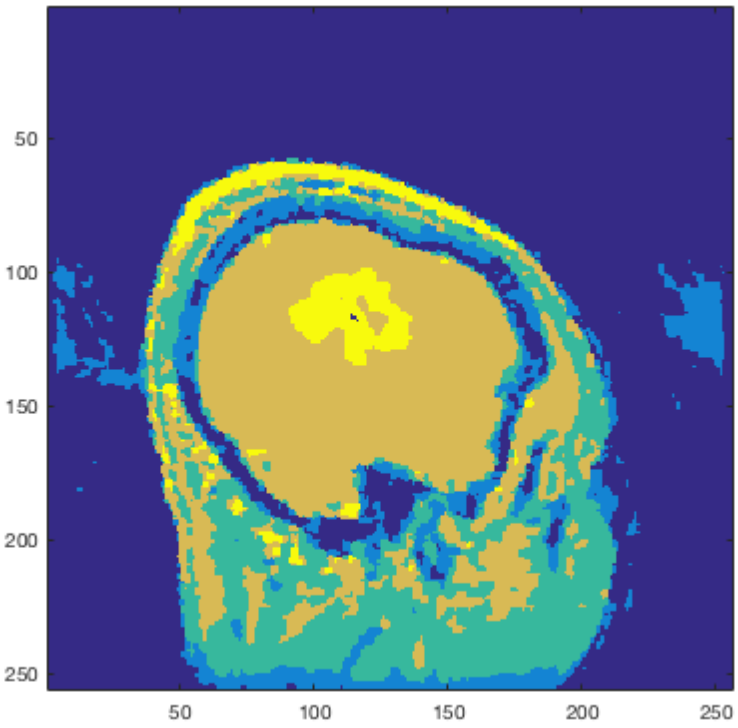
```
  figure(3*i);
  imagesc(K);
  colormap('gray');
  axis('image');
  nmeans{i}
end

%
%================================ segmentB ================================
```

```
ans =

    7.3487   35.7866   79.4104  121.9027  169.7822


ans =

    5.2687   15.5242   45.8232   80.0602  111.4166
```

Original Westin Image



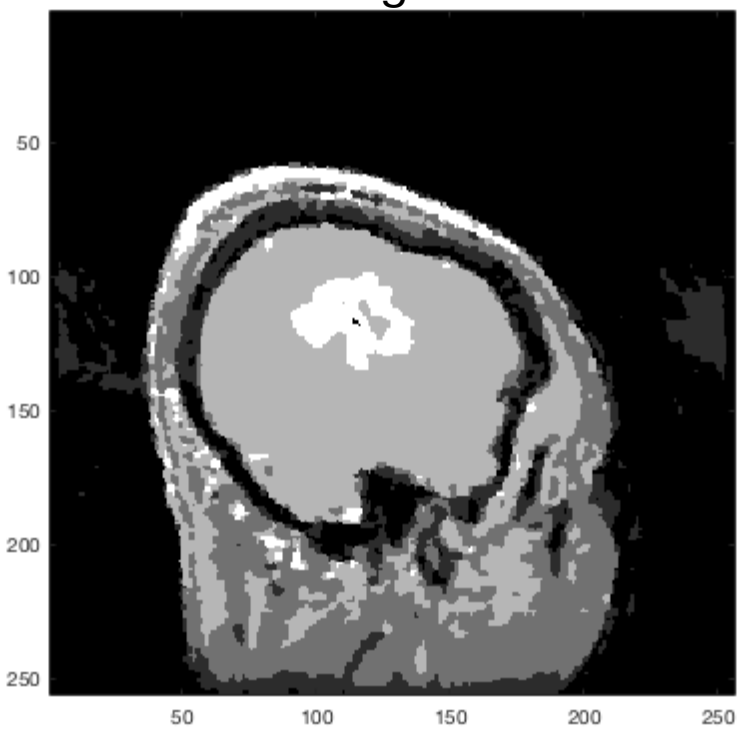
Bayesian Relaxation Algorithm Image Segmentation for Westin Image



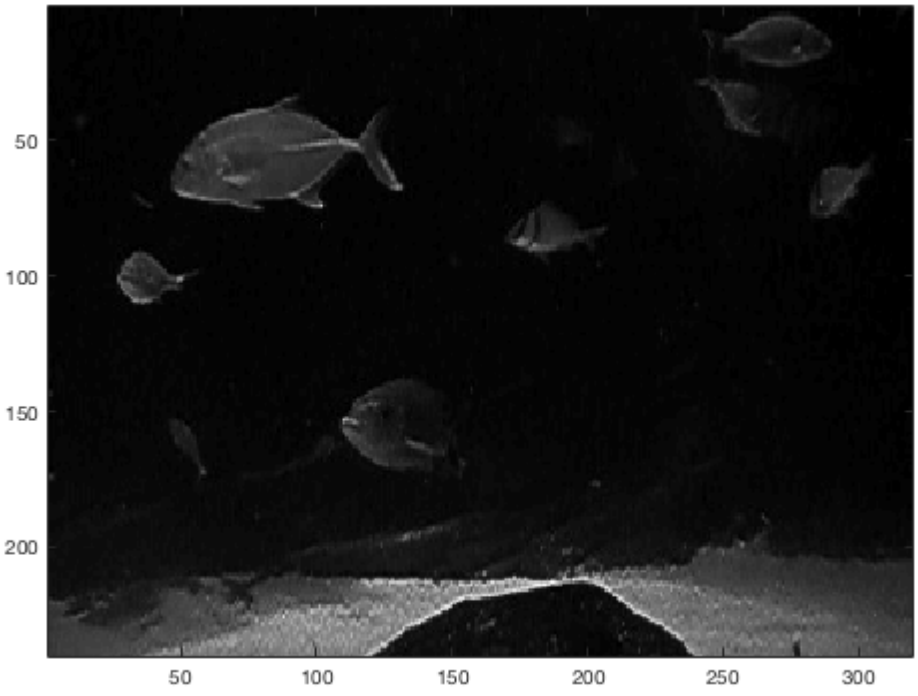
Iterations: 10
Initial Means: [3, 10, 60, 140, 150]
Standard Dev: 1

The Bayesian Relaxation Algorithm produces a similar result to the ICM Algorithm, as the two images are nearly identical. If the Bayesian algorithm was allowed to continue to convergence, the segments would be smoother. Please see page 9 for the ICM images.

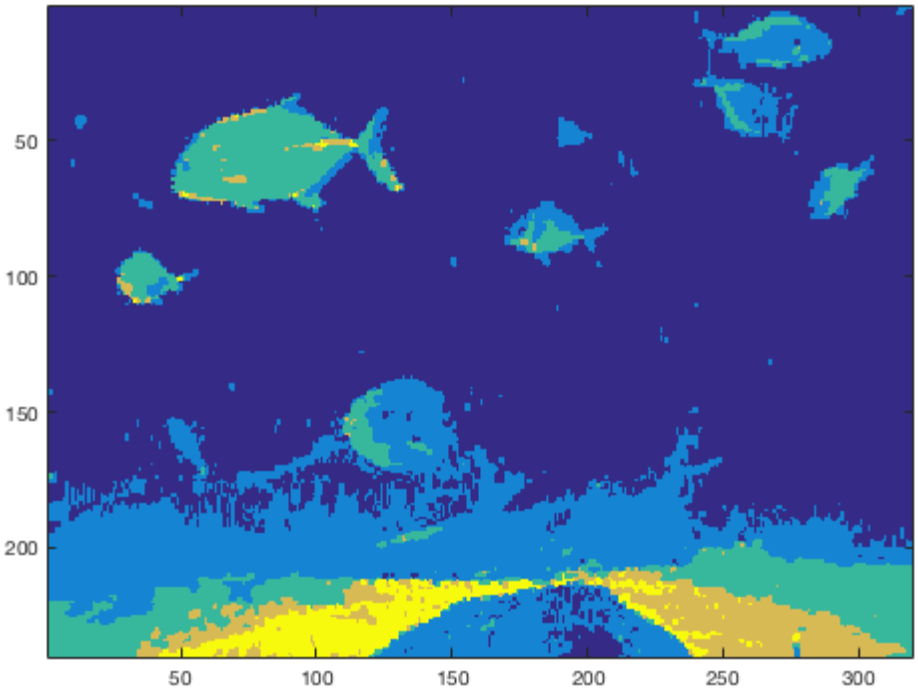
Bayesian Relaxation Algorithm Average Image Value for Westin Image



Original Fish04 Image

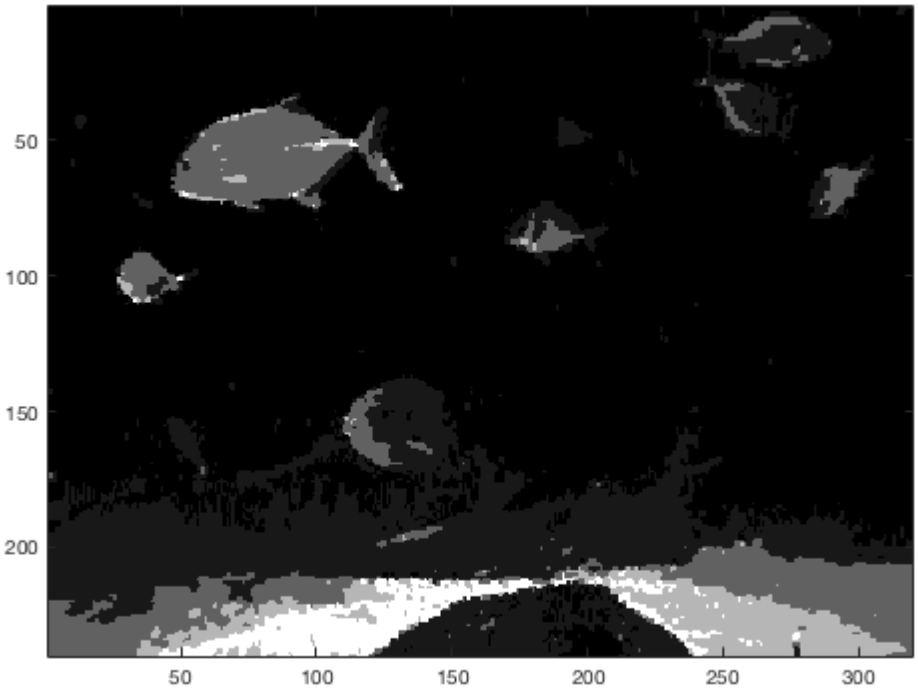


Bayesian Relaxation Algorithm Image Segmentation for Fish04 Image



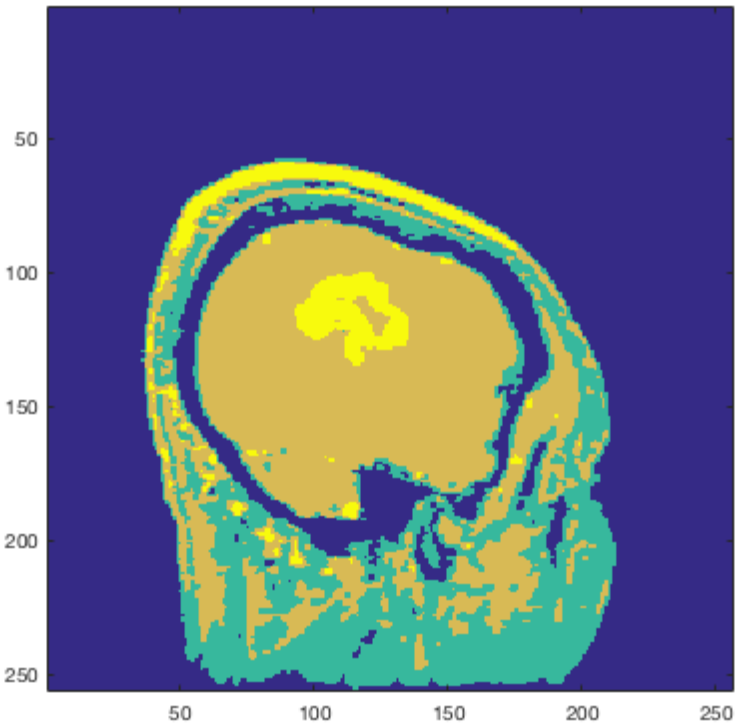
Iterations: 10
Initial Means: [1, 5 ,40, 80, 120]
Standard Dev: 1

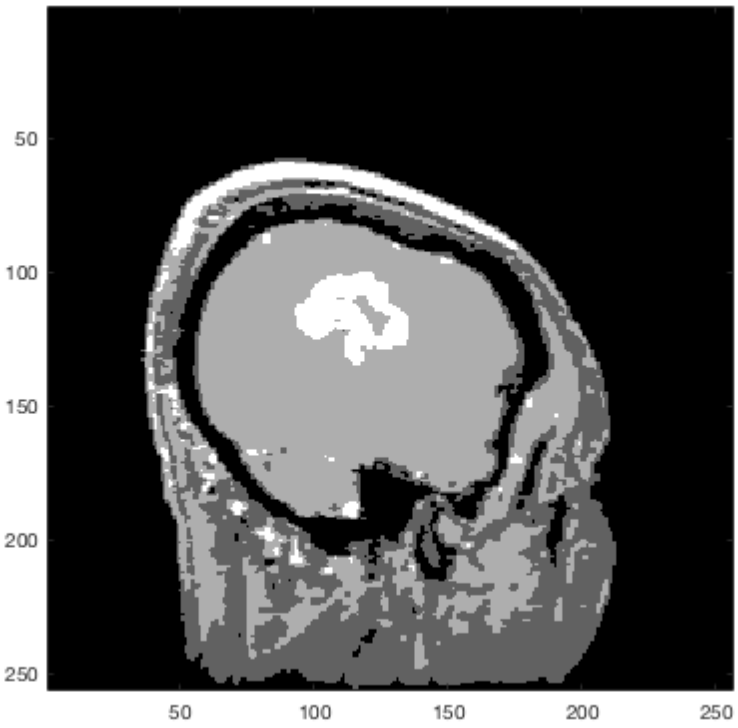
Bayesian Relaxation Algorithm Average Image Value for Fish04 Image

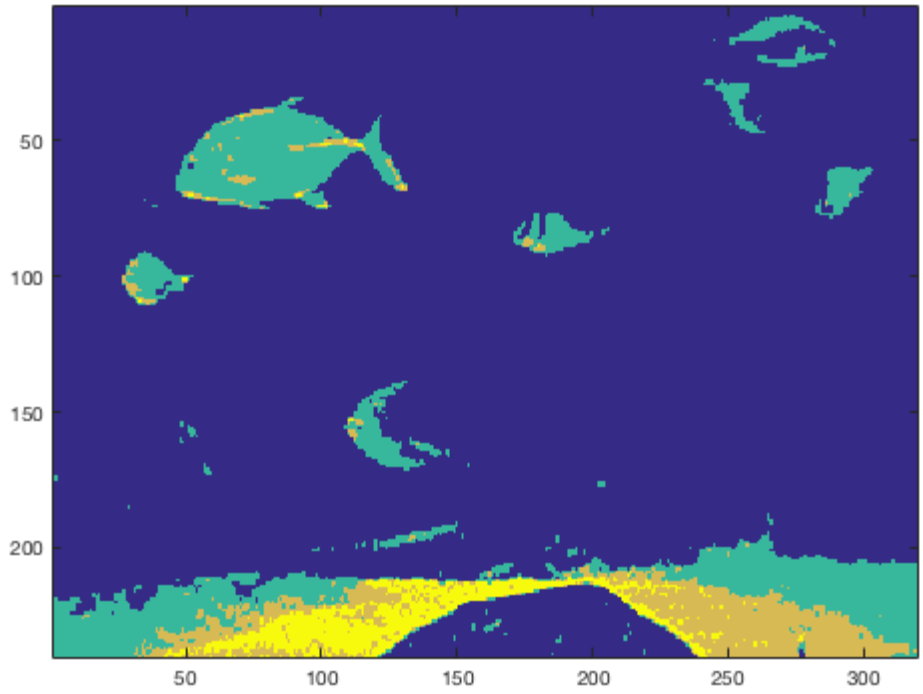


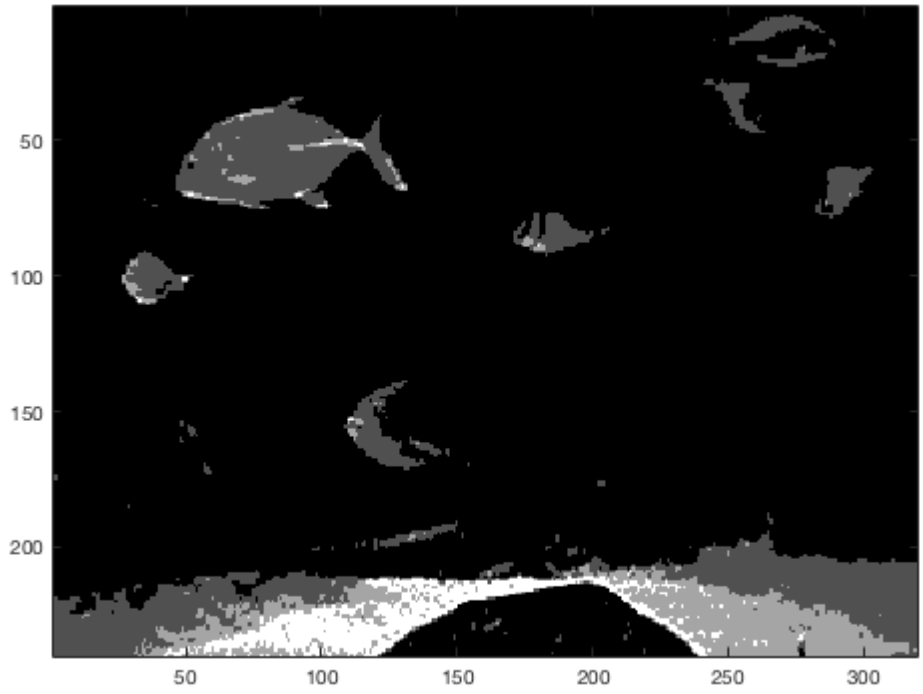
Similar to the westin image, Bayesian Relaxation has a similar outcome for the fish04 image as the ICM segmentation algorithm. There is more of a difference between this image and that of ICM, as the background has not converged to a single segment. This allows more of the fish features to be distinguished from the background. Please see page 10 for the ICM images.

Published with MATLAB® R2016b


50
100
150
200
250
50
100
150
200
250


50
100
150
200
250
50
100
150
200
250





Published with MATLAB® R2016b