```
function g = pool_adjacent_violators(h,inc)
  % Compute the isotonic regression of an histogram h.
  % inc = boolean value saying if we want the non-decreasing (inc = 1) or decreasing regression
  if (inc == 0)
    g = h;
    for i = 1 : length(h)
      som = g(i);
      for j = i-1 : -1 : 1
        if(j == 1 || (g(j)*(i-j) >= som))
          som = som / (i-j);
          for k = j+1 : i
            g(k) = som;
          end
          break;
        end
        som = som + g(j);
      end
    end
  end
  if (inc == 1)
    g = h;
    for i = length(h)-1 : -1 : 1
      som = g(i);
      for j = i+1 : length(h)
        if(j == length(h) || (g(j)*(j-i) >= som))
          som = som / (j-i);
          for k = i : j-1
            g(k) = som;
          end
          break;
        end
        som = som + g(j);
      end
    end
  end
end


function max_entrop = max_entropy(h,a,b,e,inc)
  % Compute the maximum entropy of the histogram h(a:b) for the increasing or decreasing hypothesis
  % inc = boolean value indicating if we test the increasing or
  % decreasing hypothesis
  % h = histogram
  % e = parameter used to compute the entropy
  g = h(a:b);
  decreas = pool_adjacent_violators(g, inc);
  L = length(g);

  % integrate signals
  g       = cumsum(g);
  decreas = cumsum(decreas);

  % meaningfullness threshold
  N = g(L);
  seuil = (log(L*(L+1)/(2)) + e*log(10)) / N;

  % search the most meaningfull segment (gap or mode)
  max_entrop = 0.;
  for i = 1 : L
    for j = i : L
      if (i == 1)
        r=g(j);
      else
        r = g(j) - g(i-1);
      end
      r = r / N;
      if (i == 1)
        p = decreas(j);
      else
        p = decreas(j) - decreas(i-1);
      end
      p = p / N;
      v = entrop(r, p);
      if (v > max_entrop)
        max_entrop=v;
      end
    end
  end
  max_entrop = (max_entrop - seuil)*N;
end


function v = entrop(x,y)
  % function computing the entropy between x and y. x and y must be in the interval [0,1]
  if (x == 0.)
```

```
    v = -log10(1-y);
  elseif (x == 1.0)
    v = -log10(y);
  else
    v = (x*log10(x/y) + (1.0-x)*log10((1.0-x)/(1.0-y)));
end


%%%%%%%%%%%%%%%%%
% The following functions implement the Fine to Coarse Histogram Segmentation described in
% J. Delon, A. Desolneux, J-L. Lisani and A-B. Petro, A non parametric approach for histogram segmentation,
% IEEE Transactions on Image Processing, vol.16, no 1, pp.253-261, Jan. 2007.
% Usage :
%   u = double(imread('../images/lena.png'));
%   H = hist(u(:),0:255);
%   idx=FTC_Seg(H,0);
%   idx should contain the list of all minima separating the modes of H
%%%%%%%%%%%%%%%%%
function idx = FTC_Seg(H,e)
  % FTC_seg
  % H = the integer-valued histogram to be segmented.
  % e = parameter of the segmentation
  % (corresponds to e = -log10(epsilon) in the paper)
  % large e => coarse segmentation
  % small e => fine segmentation
  lH = length(H);

  %% find the list of local minima and maxima of H
  [p, idx_max] = findpeaks(H);
  [p, idx_min] = findpeaks(-H);
  idx = sort([idx_min, idx_max]);
  if (idx(1) ~= 1)
    idx = [1, idx];
  end
  if (idx(end) ~= lH)
    idx = [idx, lH];
  end

  % find if idx starts with a minimum or a maximum
  if H(idx(1)) < H(idx(2))
    begins_with_min =  1;
  else
    begins_with_min =  0;
  end

  %% FILL THE LIST OF ENTROPIES FOR ALL MODES
  % The merging of two contiguous modes [a,b] and [b,c] can be done in two ways,
  % either by using the maximum M1 on [a,b] and by testing the decreasing hypothesis on [M1,c],
  % or by using the maximum M2 on [b,c] and by testing the increasing hypothesis on [a,M2].
  % For each configuration, we compute the entropy of the worst interval against the considered hypothesis.
  K = length(idx);
  val=zeros(1, K-3);

  % Loop on all optimas
  for k = 1 : K-3
    % decide if we want to test the increasing or decreasing hypothesis on [idx(k),idx(k+3)]
    if ((begins_with_min && mod(k,2) == 1) | (~begins_with_min && mod(k,2) == 0))
      inc = 1;
    else
      inc = 0;
    end
    % compute the max entropy on the interval [k,k+3]
    val(k) = max_entropy(H, idx(k), idx(k+3), e, inc);
  end

  %%%%%%%% MERGING of MODES
  [valmin, kmin] = min(val);      % [idx(kmin), idx(kmin+3)] is the first interval to merge
  while(~isempty(val) && valmin<0)
    % update the list of min, max
    idx = [idx(1 : kmin), idx(kmin+3 : end)];
    val = [val(1 : min(kmin,end)), val(kmin+3 : end)];
    val = val(1 : length(idx)-3);

    % update max_entropy around the removed optima
    for j = max(kmin-2, 1) : min(kmin, length(val))
      % decide if increasing or decreasing
      if (begins_with_min && mod(j,2) == 1) | (~begins_with_min && mod(j,2) == 0)
        inc = 1;
      else
        inc = 0;
      end
      % update the max entropy on the interval [k,k+3]
      val(j) = max_entropy(H, idx(j), idx(j+3), e, inc);
    end
    [valmin, kmin] = min(val);
```

```
  end

  if (begins_with_min)
    idx = idx(1 : 2 : end);
  else
    idx = idx(2 : 2 : end);
  end

  %% Display the segmentation
  bar(H, 'r');
  hold on;
  for k = 1 : length(idx)
    line([idx(k) idx(k)], [0 max(H(:))]);
  end
  hold off;
end
```