Submission before: 23.11.2015
Discussion on: 24.11.2015
Submission on stud.ip, submission folder for sheet.
Please submit a zip file containing the .m files for Matlab programming tasks.

Exercise 1 *(region labeling – 4p)*

(a) Implement a function that performs region labeling:

```
function [labels, count] = label_regions(bw_img)
%LABEL_REGIONS Perform region labeling.
%    LABEL_REGIONS(BW_IMG) Label the regions in the binary
%    (black and white) image BW_IMG. The function returns
%    an indexed image LABELS of the same size, containing
%    the region labels. COUNT holds the number of regions
%    discovered.

     % put your code here ...
end
```

Obviously, the use of `bwlabel` and similar functions is not allowed here!

(b) Apply the function to the image `segments.png` from the lecture. First convert it into a binary image. Then apply your function to it and display the result (you may use `label2rgb` here). Then try different types of postprocessing (you may use MATLAB functions like `imopen` and `medfilt2` here): (1) apply an opening to remove small regions, (2) apply a median filter (3) remove regions with less then n pixels. Compare the results.

Exercise 2 *(relaxation labeling – 8p)*

Implement the relaxation labeling algorithm from the lecture. Proceed in four steps:

(a) Write a function that computes the initial label probabilities P^0, given a label image (for the initial probability use the values from the lecture):

```
function probs = initialize_probabilities(labels,p)
%INITIALIZE_PROBABILITIES Initialize relaxation labeling.
%    PROBS = INITIALIZE_PROBABILITIES(LABELS,P) Initialize the
%    probability values for relaxation labeling. LABELS is an
%    indexed image holding region labels. P is the probability
%    that should initially be assigned to the given labels.
%    PROBS is a three dimension array of size
%    SIZE(LABELS) x NUMBER_OF_LABELS, holding the initial
%    probabilities for every label.

     % put your code here ...
end
```

(b) Write a function that performs the update step $P^n \rightarrow P^{n+1}$ (again use the default values from the lecture here):

```
function new_probs = update_probabilities(probs)
%UPDATE_PROBABILITIES Initialize relaxation labeling.
%    [NEW_PROBS CHANGE] = UPDATE_PROBABILITIES(PROBS)
%    Update the probability values using the formulae from the
```

```
%     lecture. NEW_PROBS are the updated label probabilities.

      % put your code here ...
end
```

(c) Write a function that converts the label probabilities into a label image and a confidence map.

```
function [labels, conf] = labels_for_probabilities(probs)
%LABELS_FOR_PROBABILITIES Initialize relaxation labeling.
%     [LABELS, CONF] = LABELS_FOR_PROBABILITIES(PROBS)
%     Get the LABELS from a probability matrix PROBS. Also
%     compute a confidence map CONF for these labels.

      % put your code here ...
end
```

(d) Write a loop to repeat the update step until there is no significant change. After every step display the label image along with the confidence map.

Apply your function to the binary (two labels) and the multi-labeled version of the image from exercise 1.

Exercise 3 *(watershed transform – 8p)*

Implement the watershed transform as described in the lecture.

(a) Implement the watershed transform (obviously, the use of `watershed` and similar MATLAB functions is not allowed here). Create an animation by showing the current labels after every flooding step (you may use a short `pause` to allow MATLAB to display the image). Finally, compare your result with the outcome of the MATLAB function `watershed` (e.g. for the image `dist_circles.png`).

(b) Use watershed transform to find your way through a maze (you may now use buildin functions): first transform the maze into a topographic map by applying some distant transform. Then flood that map using watershed transform. The watershed should show you the way through the maze. Display your solution.