Image Processing : Assignment-2

Digdarshan Kunwar, Eaindra Wun Pyae

15 March 2021

Contents

1 Watershed algorithm

1.1 Introduction

The watershed transform is one of the segmentation method that separates different sections within an image. It comes from the the geological watershed which are elevated terrain that separates neighboring drainage basins. The watershed transformation treats the image it operates upon like a topographic map, with the brightness of each point representing its height, and finds the lines that run along the tops of ridge. The basic idea consists of placing a water source in each regional minimum in the relief, to flood the entire relief from sources, and build barriers when different water sources meet. The resulting set of barriers constitutes a watershed by flooding.

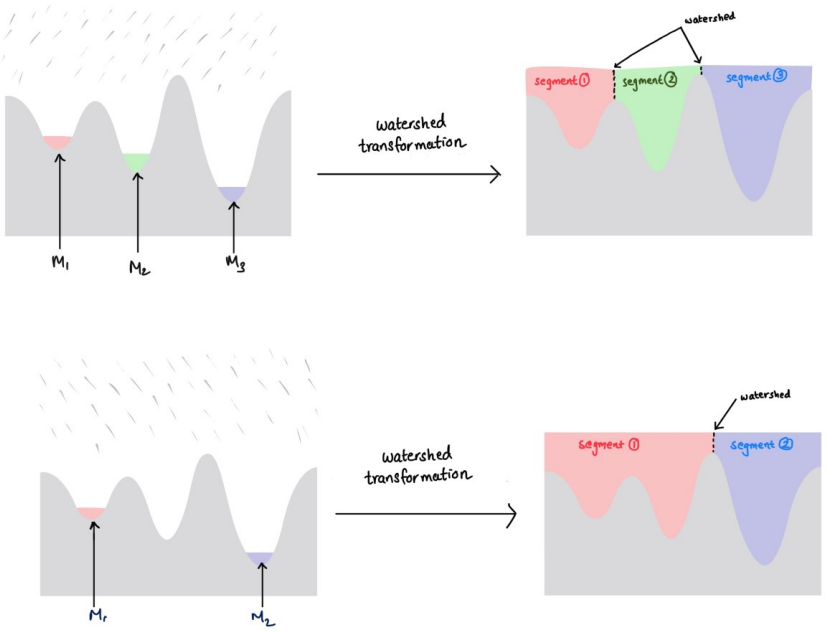


Figure 1: General watershed concept

1.2 Formal Definition

Definition in terms of flooding stimulation:
We know that catchment basin associated with a minimum **M** is denoted by CB(M). The points of this catchment basin which have an altitude less than or equal to h are denoted by $CB_h(M)$:

$$CB_h(M) = CB(M) \cap T_{i \leq a}(f)$$

After stimulating the floods in the catchment basins are progressively expanding. The image points that are first reached by the water are the one that have the least grayscale value which belong to a regional minima of the image at h_{min} where:

$$X_{h_{min}} = RMIN_{h_{min}}(f)$$

Depending on the inclusion relation with the expansion of catchment basin the regions can be defined as the unique geodesic influence zone. The recursion formula given below holds for all levels h. As the set $X_{h_{max}}$ is equal to the set of catchment basins of a grey scale image f as soon as all the levels have in flooded.

$$\begin{aligned} X_{h_{min}} =& T_{h_{min}}(f) \\ \forall h \in [h_{min}, h_{max} - 1),\ X_{h+1} =& RMIN_{h+1}(f) \cup IZ_{T_{t \leq h+1}(f)}(X_h) \end{aligned}$$

2 Implementation

Following is a pseudo-code for the flooding simulation. The data structure used is a First-In-First-Out (FIFO).

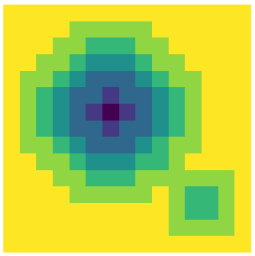
Algorithm 1 Fast watersheds

procedure WATERSHED ALGORITHM
  define $mask \leftarrow -2$
  define $wshed \leftarrow 0$
  define $init \leftarrow -1$
  define $inqueue \leftarrow -3$
  • input: f_i
    output: f_o
  • ***Initialisation***:
  -Value *init* is assigned to each pixels of $f_o : \forall p \in D_{im_o}, f_o(p) \leftarrow init$
  -$current\_label \leftarrow 0$
  -$flag : Booleanvariable$
  -$N_g(p) is the set of neighbours of p$
  • ***Sort pixels***:
  Sort the pixels of f_i in the increasing order of their grey value.
  • ***Geodesic SKIZ of levels***:
  for $h \leftarrow h_{min}$ *to* h_{max}: **do**
    ∀pixel p such that$f_i(p) = h$
    $f_o(p) \leftarrow mask$
    if $p^{'} \in N_g(p) | f_o(p^{'}) > 0$ **or** $f_o(p^{'}) >= wshed$ **then**
      $f_o(p) \leftarrow inqueue$
      `fifo_add`(p)
  while `fifo_empty()=false` **do**
    $p \leftarrow$ `fifo_retrieve()`
    for all $\forall p^{'} \in N_g(p)$ **do**
      if $f_o(p^{'} > 0$ **then**
        if ($f_o(p)$=*inqueue* or ($f_o(p) = wshed$ and $flag = true$)) **then**
          $f_o(p) \leftarrow f_o(p^{'})$
        else
          if ($f_o(p) > 0$ and $f_o(p) \neq f_o(p^{'}$ **then**
            $f_o(p) \leftarrow wshed$
            $flag \leftarrow false$
      else
        if $f_o(p^{'}) = wshed$ **then**
          if ($f_o(p) = inqueue$) **then**
            $f_o(p) \leftarrow wshed$
            $flag \leftarrow true$
        if $f_o(p^{'}) = mask$ **then**
          $f_o(p^{'}) \leftarrow inqueue$
          fifo_add($p^{'}$)
    for ∀ pixel p such that $f_i(p) = h$ **do**
      if $f_o(p) = mask$ **then**
        $current\_label \leftarrow current\_label$
        $fifo\_add(p)$
        $f_o(p) \leftarrow current\_label$
        while fifo_empty()=false **do**
          $p^{'} \leftarrow$ `fifo_retrieve()`
          for all $pixel p^{''} \in N_g(p^{'})$ **do**
            if $f_o(p^{''}) = mask$ **then**
              $fifo\_add(p^{''})$
              $f(p^{''}) \leftarrow current\_label$

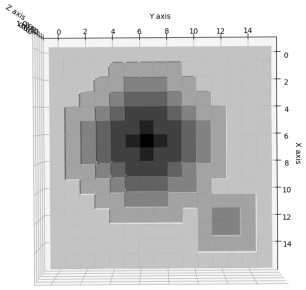
Gray-scale images are also considered to be topographical reliefs in the field of image analysis and mathematical morphology. In the topographic representation of the given image, the numerical value (i.e. the gray tone) of each pixel is the elevation at this point. Such a representation is incredibly useful, as it first enables one to properly understand the effect of a given transformation on the image under analysis.
For eg, the opening eliminates some of the peaks and crest lines, while the closing continues to fill the basins and valleys. Furthermore, thanks to this representation, concepts such as minima, catchment basins and watersheds can be well established for grayscale pictures. Here we have the topographical representation of the input images for greater intuition about images on its topographical representations.

2.1 Analysing input images

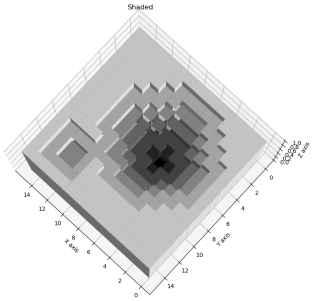
Input image f1_dinv



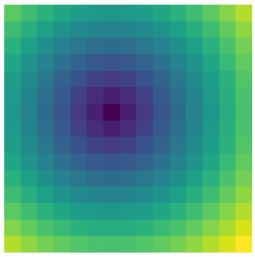
(a) f1 original image



(b) topographical representation



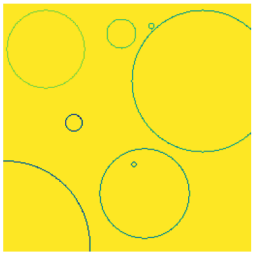
(c) topographical representation



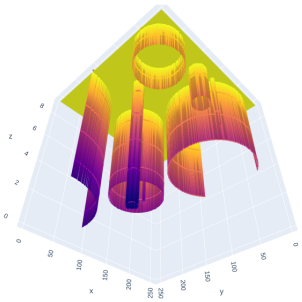
(d) DT of f1_dinv

Figure 2: f1_dinv images

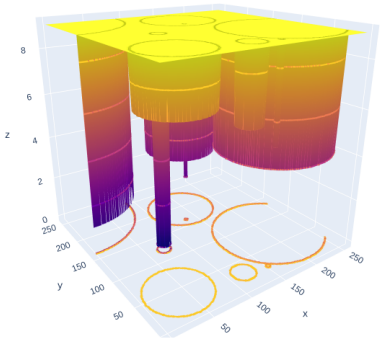
Input image f2



(a) f1 original image



(b) topographical representation



(c) topographical representation

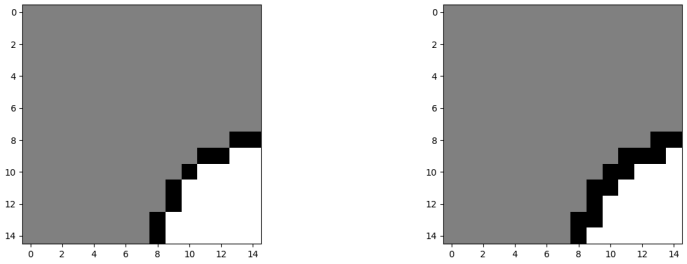
Figure 3: Images of f2

2.2 Assignment Tasks

The watershed algorithm as described as pseudo code in the attached pdf for 4-connected and 8-connected neighborhood was implemented and the results that were generated from the program are analysed below.

2.2.1 Distance transformation

The watershed result with 4/8 connected neighbors to the inverse of the distance transform $'f1\_dinv.txt'$ derived from a blob image is as indicated in Figure 4.



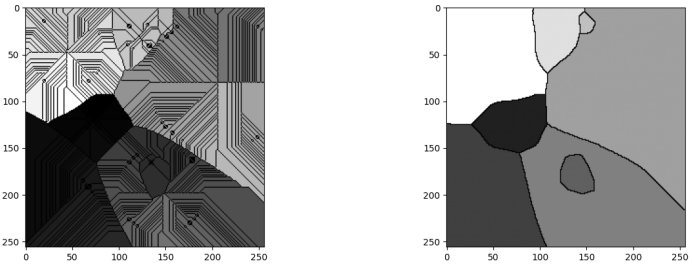
(a) With 4 connected neighbors (b) With 8 connected neighbors

Figure 4: Watershed of f1_dinv

2.2.2 4 vs 8 neighbourhood

The circle image (f2.txt). For this image, explain the differences between the 4 and 8-connected neighborhood implementations. Hint: 8-connected should produce much less regions - why?

The watershed result with 4/8 connected neighbors to the circle image f2 is indicated in Figure 5.



(a) With 4 connected neighbors (b) With 8 connected neighbors

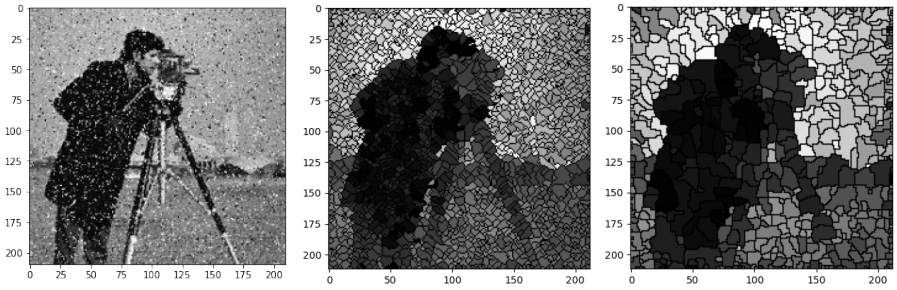
Figure 5: Watershed of f2

In Figure 5, we can see that over 300 clusters are created when we check only 4 connected neighbors of each pixel, whereas only 9 clusters are created when we check each pixel with 8 connected neighbors. The reason is that a greater connectivity between pixels is formed with more connected neighbors during the iteration steps. More neighbor pixel coordinates are added to the queue (8 rather than 4), so more similar pixels are gathered in the same cluster.

2.2.3 Further Images

Three further images of your choice that shows multiple objects each to be segmented. Before applying the watershed transform, you must derive a suitable segmentation function through a method of your choice. This could include, for example, morphological gradient computation based on erosions and/or dilations, noise removal (e.g. median filter) or smoothing (e.g. bilateral or binomial filter) of the image before and/or after gradient computation, or a marker based approach with homotopy modification through morphological reconstruction, etc.

1. **The original cameraman image**
   The direct application of the watershed transform results in a highly over-segmented scene, as seen in 6. To fight over-segmentation, we use the balanced filter that uses both median and mean filters, once more to maintain the edges while reducing noise. We see that in 6b we have a lot of small segments which are reduced in as the neighbors increase. Also as we pre-process to remove noise in both and both of the images have relatively better results than without any smoothing filters.

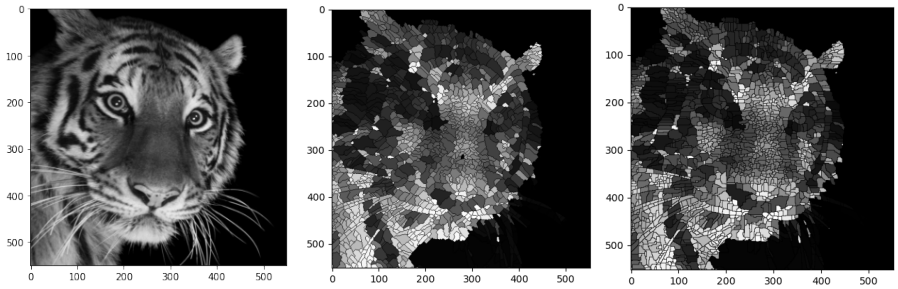


(a) Original image (b) 4-neighbors-balanced (c) 8-neighbors-median

Figure 6: Watershed of cameraman image

2. **Tiger Image**
   Segmentation of images plays a key role in machine learning, especially in applications for the classification of images. In this experiment, we are trying to segment the a tiger which could potentially help in animal recognition using deep learning algorithms.

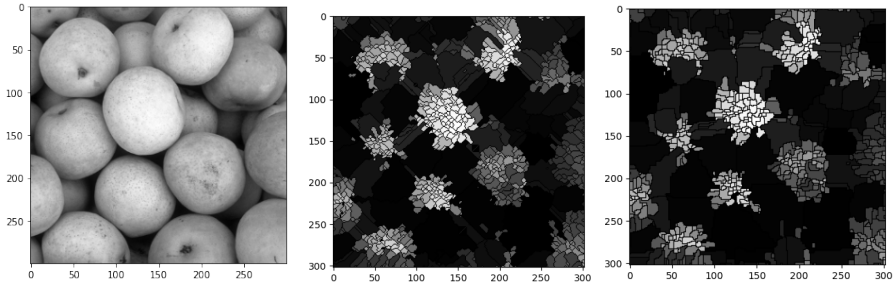
   Again, Figure 7 tells us that the direct implementation of the watershed transforms the picture into a highly over-segmented image. In order to combat over-segmentation, we use the median and balanced filter to maintain the edges while reducing the noise.



(a) Original image (b) 4-neighbors-median (c) 8-neighbors-balanced

Figure 7: Watershed of image with tiger

3. **Potatoes Image**



(a) Original image (b) 4-neighbors-balanced (c) 8-neighbors-balanced

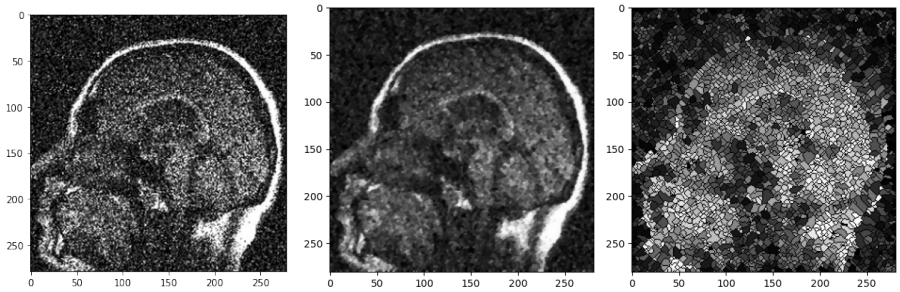
Figure 8: Watershed of potatoes image

2.3 Experimentation with further images

These are the further experimentation.

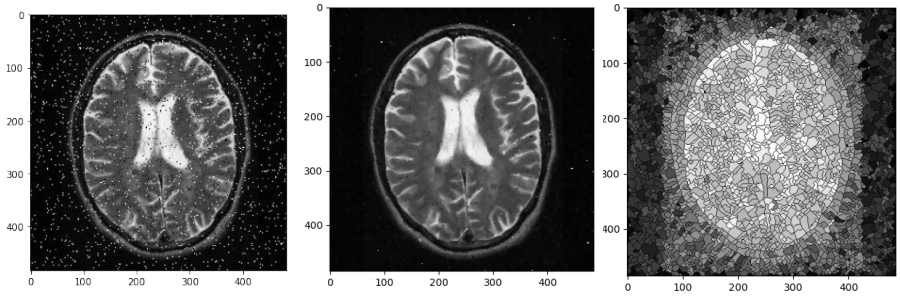
2.3.1 Medical Imaging

Segmenting MRI slices of the brain is another useful use of segmentation functions in medical image processing. This is needed to extract various tissues and organs within the brain in order to look for any irregularities related to a person's medical condition. Changes in volume, form, and regional distribution may be used to detect these anomalies. Furthermore many different modalities (X-ray, CT, MRI,Endoscopy, OCT etc) are used to create medical images. These resulting segmentations can then be used to obtain further diagnostic insights. Simple organ size estimation, cell counting, and simulations based on the extracted boundary information are all possibilities.



(a) Original image of MRI (b) MRI after noise removal (c) 4-neighbors-balanced

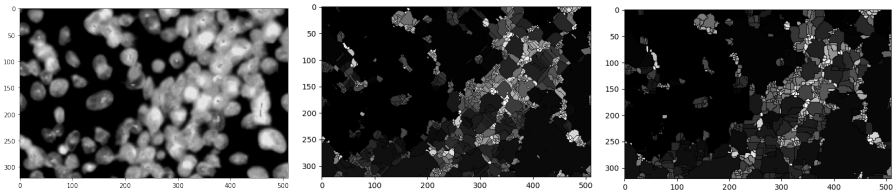
Figure 9: Watershed of MRI image



(a) Original image of brain (b) After noise removal (c) 4-neighbors-median

Figure 10: Watershed of Brain scan image

Splitting a microscopic image domain into segments that reflect individual instances of cells is known as cell segmentation. It is considered as a pillar of image-based biomedical science and is a key phase in many biomedical studies. A well-segmented picture can capture biologically important morphological information, and cellular morphology is an indication of a cell's physiological state.



(a) Original image of cells (b) 4 neighbour watershed (c) 8 neighbour watershed

Figure 11: Watershed of cells image

3 Statement of Contribution

Further works regarding this assignments can also be found in the Jupyter Notebook file.
Digdarshan Kunwar: Worked mostly on project report, topographic representation and proof reading the algorithm with experimentation of images on watershed. Worked on filtering methods and argparse for the program.
Eaindra Wun Pyae: Implementation of sorting and watershed algorithms, generating output results for images f1_dinv and f2.
Work was done in branches 'dev_dku' and 'dev_ewu' in image-processing-github.

References

[1] Image Segmentation with Watershed Algorithm https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html

[2] Image Processing Guide https://github.com/maponti/imageprocessing_course_icmc