Week 2: Implement Gaussian and median filtering

Materials

MDSC 689.03 - W2017 Students > Image_Datasets

- headGaussian.nii.tar.gz → MRI image of the head with artificially added gaussian noise. NIfTI format.
- headSaltPepper.nii.tar.gz → MRI image of the head with artificially added salt and pepper noise. NIfTI format.
- Thorax.tar.gz → CT image of the thorax. DICOM format.

Assignment

This assignment requires you to implement the Gaussian and median filtering. Please implement both filters yourself, and do not use the corresponding VTK filters (vtkImageGaussianSmooth, vtkImageMedian3D, or anything the provides a similar level of functionality). However, you may use those additionally for comparison of the results.

First, in order to understand the differences of gaussian and median filter, apply them to the images headGaussian.nii.tar.gz and headSaltPepper.nii.tar.gz. Each image has artificially added gaussian and salt and pepper noise. Apply both filters to each image and compare the results. Which filter works best for each case?

The Thorax.tar.gz dataset was taken from the Cornell University database for lung cancer and nodule detection (https://veet.via.cornell.edu/lungdb.html). These images were acquired with low radiation dose, so that they present considerable noise artefacts. The objective of this task is to apply and compare gaussian and median filtering to the Thorax.tar.gz dataset. You can also apply both filters in any order and check your results.

- Implement gaussian filtering using python and vtk.
- Implement median filtering using python and vtk.
- Apply both filters to the headGaussian and headSaltPepper datasets and compare. Which filter generates the best visual results?
- Apply both filters to the Thorax dataset and compare. Which filter generates the best visual results?
- **Due on: Next monday at noon. Python files and screen captures must be uploaded to your Dropbox directory.**

Important considerations

- Check the code below for voxel navigation and value modification.
  for x in range(0, image.GetDimensions()[0]):
  *** for y in range(0, image.GetDimensions()[1]):***
  ***   for z in range(0, image.GetDimensions()[2]):***
  ***   voxelValue = image.GetScalarComponentAsFloat(x,y,z,0)***
  ***   image.SetScalarComponentFromFloat(x, y, z, 0, voxelValue)***
- Remember that the kernels have to be applied using values of the original, and not the resulting image. This is an option to create a copy of an image.
  copyImage = vtk.vtkImageData()
  copyImage.DeepCopy(image)
- Assume the following kernel for the gaussian filter:
  1 2 1 2 4 2 1 2 1
  2 4 2 4 16 4 2 4 2
  1 2 1 2 4 2 1 2 1
  ...or implement your own gaussian function and create your own kernel, and enjoy.
- Python can declare matrices like this:
  kernel = [[[1, 2, 1], [2, 4, 2], [1, 2, 1]], [[2, 4, 2], [4, 16, 4], [2, 4, 2]], [[1, 2, 1], [2, 4, 2], [1, 2, 1]]]
  and uses loops like this:
  for x in range(start, end):
  Note that the loop does not the evaluate the position “end”.
- Consider the case where part of your kernel is located outside of your image, carefully. You can assume the value 0 for nonexistent voxels, or check the limits of your matrices, carefully.
- Implement the filters yourself.

Some results

No spoilers.