Algorithm 1: Adaptive Thresholding

Input : v_{in} - the input volume
s_x - the neighborhood splitting factor along x-axis
s_y - the neighborhood splitting factor along y-axis
s_z - the neighborhood splitting factor along z-axis
p - the percentile we will be using to locally threshold

Output: v_{out} - the output volume

subXLen = v_{in}.shape[0]/s_x
subYLen = v_{in}.shape[1]/s_y
subZLen = v_{in}.shape[2]/s_z
for $xInc \in \{1, \ldots, s\}$ **do**
  for $yInc \in \{1, \ldots, s\}$ **do**
    for $zInc \in \{1, \ldots, s\}$ **do**
      sub = v_{in}[(xInc - 1) * subXLen: xInc * subXLen][(yInc - 1) * subYLen: yInc * subYLen][(zInc - 1) * subZLen: zInc * subZLen]
      subThresh = binaryThreshold(sub, p)
      v_{out}[(xInc - 1) * subXLen: xInc * subXLen][(yInc - 1) * subYLen: yInc * subYLen][(zInc - 1) * subZLen: zInc * subZLen] = subThresh
    end for
  end for
end for
return v_{out}