MC920/EA979 – PDI/CG
Aluno: Davi Kooji Uezono – RA: 097464

Imagens:

	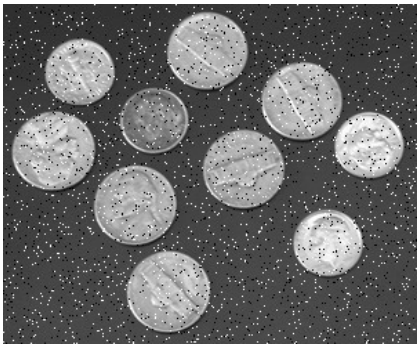	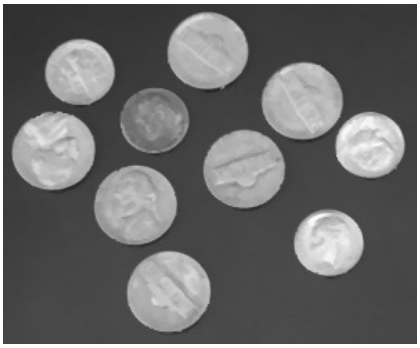
coins.pgm	coins_adding_salt_pepper.pgm	coins_removing_salt_pepper.pgm

Imagem original: http://people.sc.fsu.edu/~jburkardt/data/pgma/coins.ascii.pgm

	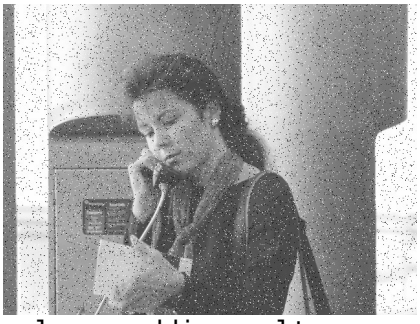	
columns.pgm	columns_adding_salt_pepper.pgm	columns_removing_salt_pepper.pgm

Imagem original: http://people.sc.fsu.edu/~jburkardt/data/pgma/columns.ascii.pgm

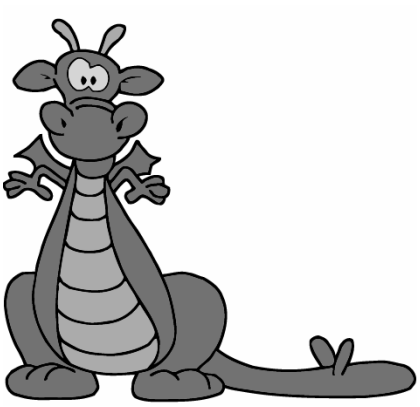	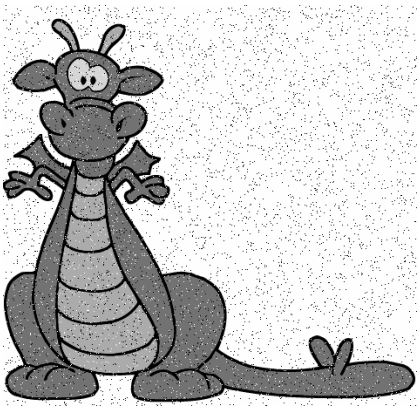	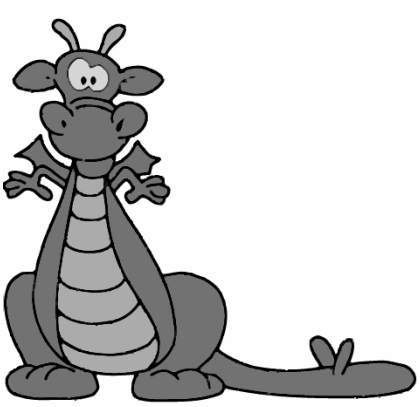
dragon.pgm	dragon_adding_salt_pepper.pgm	dragon_removing_salt_pepper.pgm

Imagem original: http://people.sc.fsu.edu/~jburkardt/data/pgma/dragon.ascii.pgm

Código fonte 1 (adicionar efeito sal e pimenta):

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from random import randint

# percentage of salt and pepper pixels
percentage = 5

name = raw_input("Please select one of these images: coins, columns or
dragon.\n")
while name not in ["coins", "columns", "dragon"]:
  name = raw_input("ERROR! Please select one of these images: coins, columns or
dragon.\n")

image = open(name + ".pgm", "r")
if image.readline() == "P2\n":
  # ignore comment line from PGM format.
  image.readline()

  size_string = image.readline()
  size = size_string.split()

  # number of pixels in image.
  n_pixels = int(size[0]) * int(size[1])
  # number of pixels in image to apply salt and pepper.
  sp_pixels = int(n_pixels * percentage / 100)

  depth_string = image.readline()
  depth = depth_string.split()
  depth = int(depth[0])

  # creating the new image with salt and pepper effect.
  salt_and_pepper = open(name + "_adding_salt_pepper.pgm", "w")
  salt_and_pepper.write("P2\n")
  salt_and_pepper.write("# created by Davi K. Uezono - RA 097464\n")
  salt_and_pepper.write(size_string)
  salt_and_pepper.write(depth_string)

  lines = image.readlines()
  for i in range(len(lines)):
    values_in_this_line = lines[i].split()
    for pixel in values_in_this_line:
      # when I have a colection of (max_prob + 1) elements, if I choose one of
      # them arbitrary, I will have the same percentage of choosing one of them
      # as defined in the very beginning of this source code.
      max_prob = int(100 / percentage) - 1
      # the random int between zero and max_prob converted to a boolean value
      # and negated will return a TRUE value in the same percentage as defined
      # in the very beginning of this source code.
      if(not bool(randint(0,max_prob)) and bool(sp_pixels > 0)):
        # arbitrary BLACK or WHITE dots in the image.
        pixel_out = str(depth * randint(0,1))
        salt_and_pepper.write(pixel_out)
        sp_pixels = sp_pixels - 1
      else:
        salt_and_pepper.write(pixel)
      salt_and_pepper.write(" ")
    salt_and_pepper.write("\n")

  image.close()
  salt_and_pepper.close()
```

Código fonte 2 (suavizar com filtro da mediana):

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from random import randint

# mask size
mask = 3

name = raw_input("Please select one of these images: coins, columns or
dragon.\n")
while name not in ["coins", "columns", "dragon"]:
  name = raw_input("ERROR! Please select one of these images: coins, columns or
dragon.\n")

salt_and_pepper = open(name + "_adding_salt_pepper.pgm", "r")
if salt_and_pepper.readline() == "P2\n":
  # ignore comment line from PGM format.
  salt_and_pepper.readline()

  size_string = salt_and_pepper.readline()
  size = size_string.split()
  n_columns = int(size[0])
  n_lines = int(size[1])

  depth_string = salt_and_pepper.readline()
  depth = depth_string.split()
  depth = int(depth[0])

  # reading the input image into a pixel list.
  pixel_list = []
  lines = salt_and_pepper.readlines()
  for i in range(len(lines)):
    values_in_this_line = lines[i].split()
    for pixel in values_in_this_line:
      pixel_list.append(int(pixel))
  salt_and_pepper.close()

  # output pixel list initialized as the same of input one.
  pixel_list_out = pixel_list
  mw = int(mask/2)

  # rastering the input image including borders.
  for i in range(n_lines):
    for j in range(n_columns):
      mask_of_pixels = []
      for x in range (i - mw, i + mw + 1):
        for y in range (j - mw, j + mw + 1):
          # ignoring negative bounds.
          if (x >= 0 and x < n_lines and y >= 0 and y < n_columns):
            mask_of_pixels.append(pixel_list[x*n_columns + y])
      # median value is the central value of an ordered vector.
      mask_of_pixels.sort()
      median = int(len(mask_of_pixels)/2)
      pixel_list_out[i*n_columns + j] = mask_of_pixels[median]

  # initializing the final (clean) image.
  clean_image = open(name + "_removing_salt_pepper.pgm", "w")
  clean_image.write("P2\n")
  clean_image.write("# created by Davi K. Uezono - RA 097464\n")
  clean_image.write(size_string)
  clean_image.write(depth_string)
```

```
# printing final pixel values in the final (clean) image.
for i in range(n_lines):
  for j in range(n_columns):
    clean_image.write(str(pixel_list_out[i*n_columns + j]) + " ")
  clean_image.write("\n")
clean_image.close()
```