Testing Kernel methods

Jieming You

10/1/22

Table of contents

Testing convolution filters

Loading packages

```
library(jpeg)     # For loading jpeg into matrix
library(grid)
library(raster)   # For plotting
library(magrittr)
```

Loading image

```
img <- readJPEG("wall.jpg", native = FALSE)
dim(img)
```

```
[1] 500 500   3
```

500 x 500 pixels, three channels per pixel

```
plot(as.raster(img))
```

2D Gaussian function is defined as

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where the x and y are delta from the pixel. For example, if we're calculating the gaussian kernel for (0, 0), kernel radius of 3 would yield an x and y ranging from -3 to 3.

Computing 2-D Gaussian Kernel

```
# w = width
# s = sigma
gaussian_kernel <- function(w, s) {
  if (w%%2 != 1) { stop("Length must be an odd number") } # Odd Check

  # Boundaries
  right  <- (w-1)/2
  left <- -right

  # Setting up the linspace
  ax <- seq(left, right, length=w)

  # Draw from gaussian cdf
  gauss <- dnorm(ax, sd = s)
```

```
  # Outer product to form a kernel matrix
  kernel <- outer(gauss, gauss)
  kernel <- kernel / sum(kernel)

  return(kernel)
}

gaussian_kernel(3, 0.5)
```

```
          [,1]      [,2]      [,3]
[1,] 0.01134374 0.08381951 0.01134374
[2,] 0.08381951 0.61934703 0.08381951
[3,] 0.01134374 0.08381951 0.01134374
```

Function for padding matrices using edge extension

```
matrix_padding <- function(mat, pad) {
  m <- ncol(mat)
  n <- nrow(mat)
  top <- c( rep(mat[1,1], pad), mat[1,], rep(mat[1,m], pad) )
  bot <- c( rep(mat[n,1], pad), mat[n,], rep(mat[n,m], pad) )
  left <- matrix(rep(mat[2:n, 1], pad), ncol=pad)
  right <- matrix(rep(mat[2:n, m], pad), ncol=pad)
  mid <- cbind( left, mat[2:n,], right)

  new_mat <- rbind( matrix(rep(top, pad), nrow=pad, byrow=TRUE),
                    mid,
                    matrix(rep(bot, pad+1), nrow=pad+1, byrow=TRUE))
  return(new_mat)
}

matrix_padding(matrix(1:9, ncol=3), pad=3) %>%
  write.table(row.names=F, col.names=F)
```

```
1 1 1 1 4 7 7 7 7
1 1 1 1 4 7 7 7 7
1 1 1 1 4 7 7 7 7
2 2 2 2 5 8 8 8 8
3 3 3 3 6 9 9 9 9
```

```
3 3 3 3 6 9 9 9 9
3 3 3 3 6 9 9 9 9
3 3 3 3 6 9 9 9 9
3 3 3 3 6 9 9 9 9
```

Function for the rolling kernel method with extended edge handling

```r
rolling_kernel <- function(source, kernel) {

  output <- source
  x <- ncol(source)
  y <- nrow(source)
  m <- ncol(kernel)
  n <- nrow(kernel)
  mmid <- (m-1)/2
  nmid <- (n-1)/2
  mmid2 <- 2*mmid
  nmid2 <- 2*nmid

  # Kernel dimensions exceed the source
  if (m > x ||  n  > y) {
    stop("Kernel length exceeds the source")
  }

  source_extended <- matrix_padding(source, mmid)

  for (i in 1:x) {
    for (j in 1:y) {
      # i + pad - pad ---> i + pad + pad
      # j + pad - pad ---> i - pad + pad
      source_kernel <- source_extended[(i):(i+mmid2), j:(j+nmid2)]
      out_kernel <- source_kernel * kernel
      out_pixel <- sum(out_kernel)
      output[i, j] <- out_pixel
    }
  }
  return(output)
}
```

Main wrapper function

```
gaussian_blur <- function(image, width, sigma) {
  kernel_gaussian <- gaussian_kernel(width, sigma)
  channels <- dim(image)[3]
  output <- image
  for (c in 1:channels) {
    output[,,c] <- rolling_kernel(image[,,c], kernel_gaussian)
  }
  return(output)
}
```

Applying the Gaussian Blur

```
img_gaussian <- gaussian_blur(img, width = 41, sigma = 3)
img_gaussian2 <- gaussian_blur(img, width = 61, sigma = 7)

# Plot
par(mfrow=c(1,3))
plot(as.raster(img))
mtext("Original image")
plot(as.raster(img_gaussian))
mtext(expression(atop(Gaussian~kernel~convolution, w~"="~41~and~sigma^2~"="~3)),
      col="red")
plot(as.raster(img_gaussian2))
mtext(expression(atop(Gaussian~kernel~convolution, w~"="~61~and~sigma^2~"="~5)),
      col="red")
```

Original image

Gaussian kernel convolution
w = 41 and $\sigma^2 = 3$

Gaussian kernel convolu
w = 61 and $\sigma^2 = 5$