Segumda Prova de Processamento de Imagens

Gabriel D'assumpção de Carvalho

2025-07-30

Bibliotecas

```
# install.packages("png")

set.seed(42)

library(png)
```

4º Questão

Introdução

Nesta atividade, trabalhamos com uma imagem em tons de cinza, adicionamos ruído gamma (com forma e escala iguais a 1) e padronizamos a imagem para o intervalo [0, 1] para facilitar o processamento. Em seguida, aplicamos a transformada de Fourier 2D com um deslocamento espectral (ffshift), centralizando a *Direct Current* (DC), e aplicamos um filtro passa-baixa.

Carregando a imagem

A imagem é carregada com a biblioteca `png`. Como possui três canais (RGB), foi convertida para tons de cinza pela média dos canais e padronizada para o intervalo [0, 1].

```
getwd()
```

[1] "/home/gabrieldadcarvalho/github/image_processing/exam/2_exam"

```
img <- readPNG("my_picture.png")
N <- dim(img)[1]
print("Dimensões da imagem:")
```

[1] "Dimensões da imagem:"

```
dim(img)
```

[1] 50 50 3

```
# Padronizando a imagem para [0, 1]
img <- (img - min(img)) / (max(img) - min(img))
```

```
plot(1:2, type = "n", xlab = "", ylab = "")
rasterImage(img, 1, 1, 2, 2)
```


```
imGray <- (img[, , 1] + img[, , 2] + img[, , 3]) / 3

print("Dimensões da imagem em tons de cinza:")
```

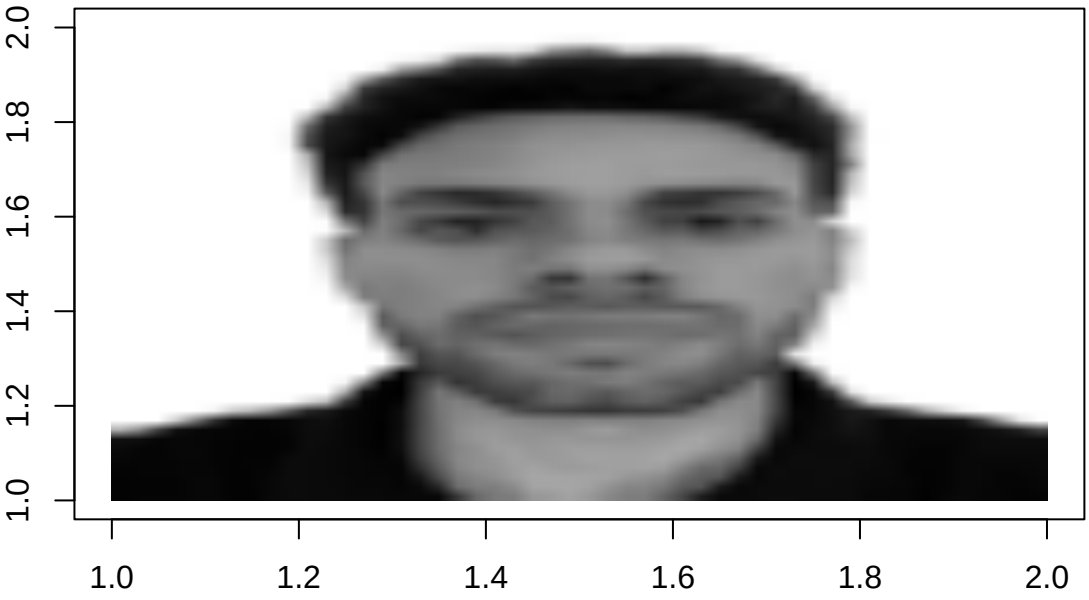
[1] “Dimensões da imagem em tons de cinza:”

```
dim(imGray)
```

[1] 50 50

```
# Padronizando a imagem para [0, 1]
imGray <- (imGray - min(imGray)) / (max(imGray) - min(imGray))
```

```
plot(1:2, type = "n", xlab = "", ylab = "")
rasterImage(imGray, 1, 1, 2, 2)
```



Adição de ruído gamma

Adicionamos ruído gamma(1, 1) à imagem em tons de cinza e novamente realizamos a padronização. O ruído prejudica a visibilidade da imagem original, restando apenas contornos tênues.

```
rGamma <- matrix(rgamma(N * N, 1, 1), nrow = N)

imGrayG <- imGray + rGamma
print("Dimensões da imagem com ruído gamma:")
```

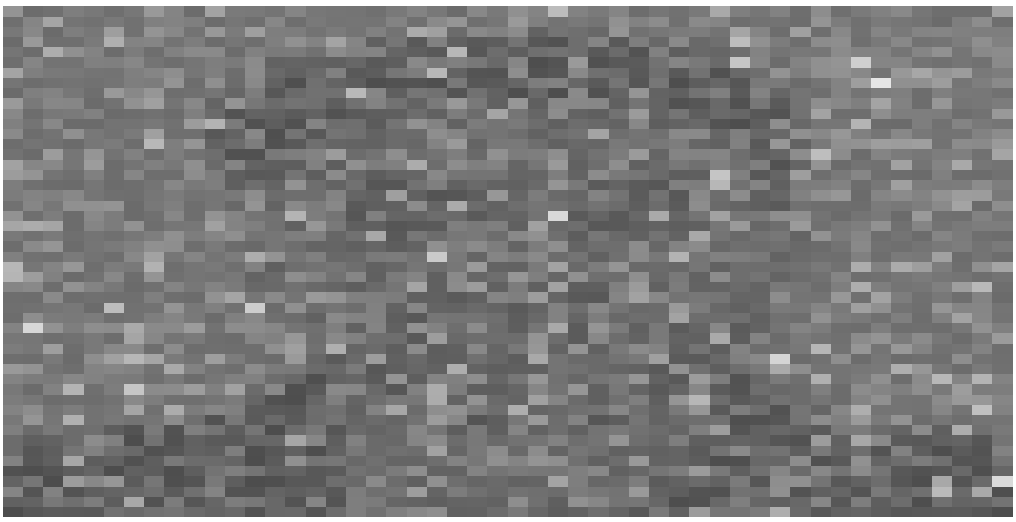
[1] “Dimensões da imagem com ruído gamma:”

```
dim(imGrayG)
```

[1] 50 50

```
# Padronizando a imagem com ruído gamma para [0, 1]
imGrayG <- (imGrayG - min(imGrayG)) / (max(imGrayG) - min(imGrayG))
```

```
image(t(apply(imGrayG, 2, rev)),
  col = gray.colors(256), axes = FALSE,
  main = "Imagem com ruído gamma"
)
```



Podemos ver que perdemos quase completamente a informação da imagem original a olho nu, sendo possível reconhecer apenas alguns contornos da imagem. Agora vamos definir e aplicar a transformada de Fourier 2d para aplicar um filtro passa-baixa.

Transformada de Fourier 2D

Ao aplicar a transformada de Fourier 2D vai ser possível notar que a energia da imagem se concentra no canto superior esquerdo. Isso é característico da componente DC (média da imagem), que foi vai ser centralizado usando o `ffshift`.

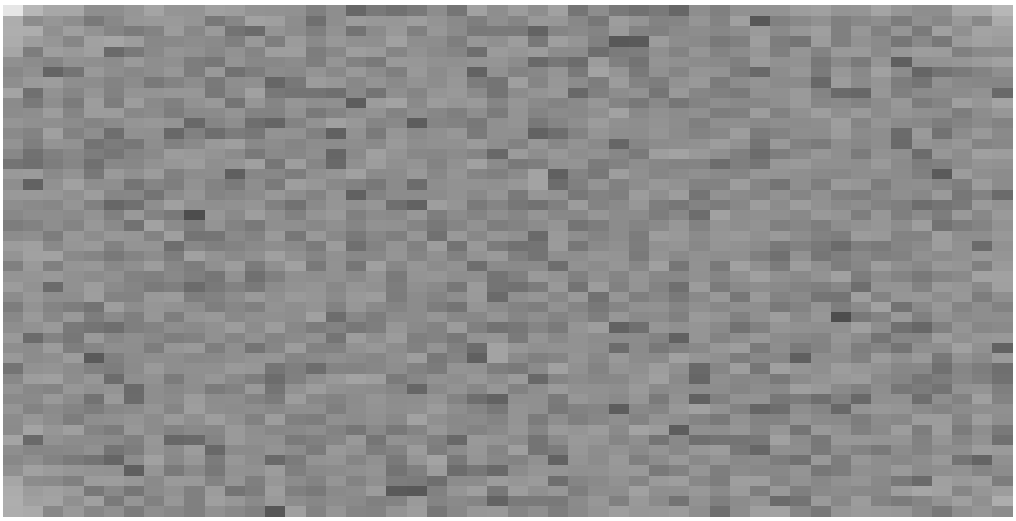
```
df2d <- function(mat) {
  # mat: Imagem
  M <- nrow(mat)
```

```
  N <- ncol(mat)
  F <- matrix(complex(real = 0, imaginary = 0), nrow = M, ncol = N)

  for (u in 0:(M - 1)) {
    for (v in 0:(N - 1)) {
      sum_val <- 0 + 0i
      for (x in 0:(M - 1)) {
        for (y in 0:(N - 1)) {
          angle <- -2 * pi * ((u * x) / M + (v * y) / N)
          sum_val <- sum_val + mat[x + 1, y + 1] * exp(1i * angle)
        }
      }
      F[u + 1, v + 1] <- sum_val
    }
  }

  return(F)
}

idf2d <- function(F) {
  # F: Matrix com transformada de fourier
  M <- nrow(F)
  N <- ncol(F)
  f <- matrix(0, nrow = M, ncol = N)

  for (x in 0:(M - 1)) {
    for (y in 0:(N - 1)) {
      sum_val <- 0 + 0i
      for (u in 0:(M - 1)) {
        for (v in 0:(N - 1)) {
          angle <- 2 * pi * ((u * x) / M + (v * y) / N)
          sum_val <- sum_val + F[u + 1, v + 1] * exp(1i * angle)
        }
      }
      f[x + 1, y + 1] <- Re(sum_val) / (M * N)
    }
  }

  return(f)
}
```

```
F <- df2d(imGrayG)
```

```
# Visualizando a transformada de Fourier 2d
image(t(apply(log(abs(F) + 1), 2, rev)),
  col = gray.colors(256),
  axes = FALSE,
  main = "Transformada de Fourier 2d"
)
```

Transformada de Fourier 2d



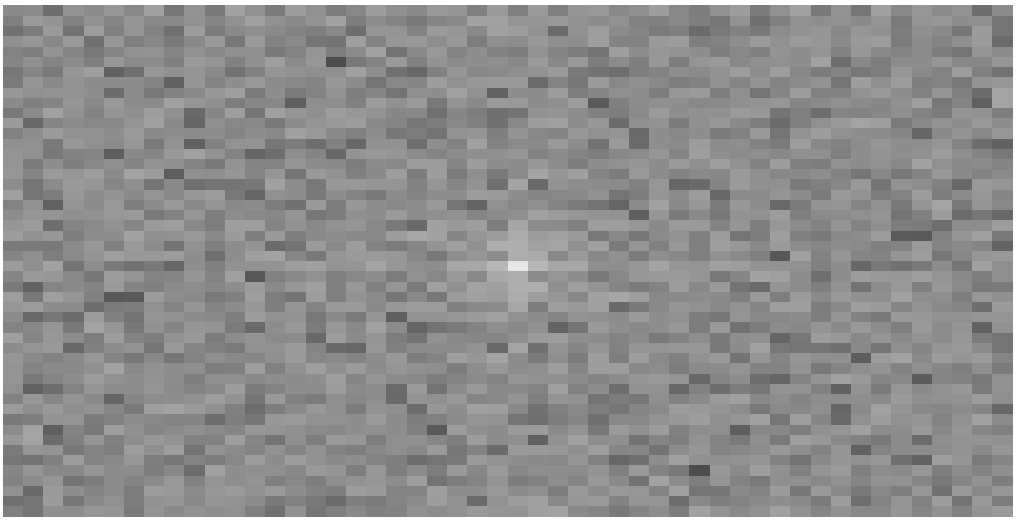
Pode-se ver que após aplicar a transformada de Fourier 2d, conseguimos ver a energia mais forte no canto superior esquerdo da imagem, o elemento (0, 0) representa a energia DC (Direct Current), que é a média da imagem. Agora vamos aplicar um ffshift para centralizar a energia DC no meio da imagem.

```
ffshift <- function(mat) {
  nr <- nrow(mat)
  nc <- ncol(mat)
  mat[c((nr / 2 + 1):nr, 1:(nr / 2)), c((nc / 2 + 1):nc, 1:(nc / 2))]
}
```

```
F_shifted <- ffshift(F)
```

```
# Visualizando a transformada de Fourier 2d com ffshift
image(t(apply(log(abs(F_shifted) + 1), 2, rev)),
  col = gray.colors(256), axes = FALSE,
  main = "Transformada de Fourier 2d com ffshift"
)
```

Transformada de Fourier 2d com ffshift

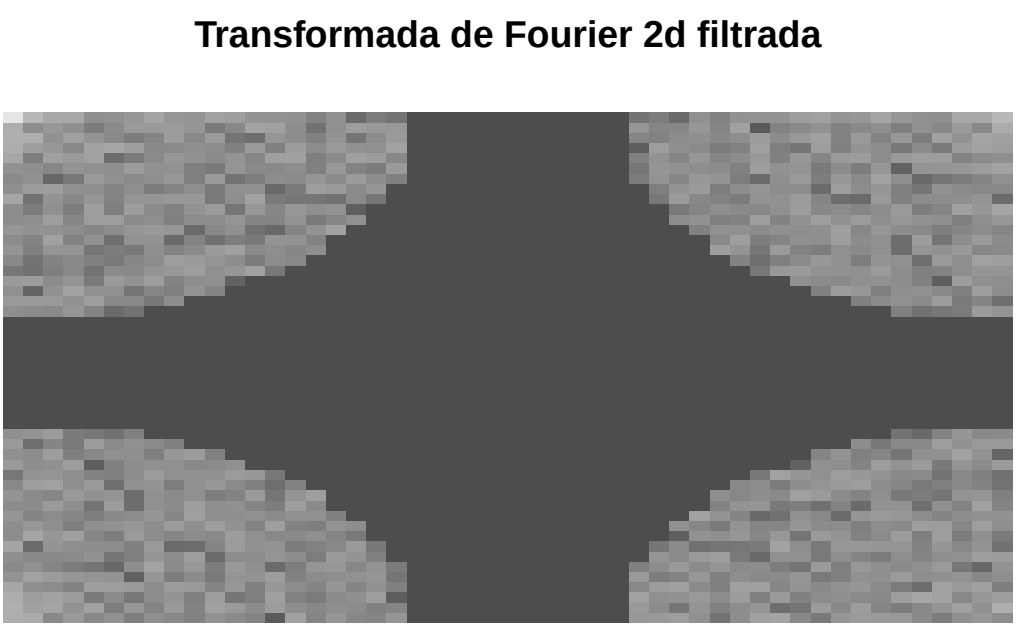


Agora vamos aplicar um filtro passa-baixa, zerando as frequências acima a uma distância de 20 pixels/raios do DC.

```
low_pass_filter <-
  function(fft_matrix, radius) {
    dim_x <- nrow(fft_matrix)
    dim_y <- ncol(fft_matrix)
    center_x <- floor(dim_x / 2) + 1
    center_y <- floor(dim_y / 2) + 1
    mask <- matrix(0, nrow = dim_x, ncol = dim_y)

    for (i in 1:dim_x) {
      for (j in 1:dim_y) {
        distance <-
          sqrt((i - center_x)^2 + (j - center_y)^2)
        if (distance < radius) {
          mask[i, j] <- 1
        }
      }
    }

    mask <- as.complex(mask)

    return(mask)
  }
```

```
radius <- 20
mask <- low_pass_filter(F_shifted, radius)

F_filtered <- F_shifted * mask

# Desfazer o ffshift
F_filtered <- ffshift(F_filtered)
```

```
# Visualizando a transformada de Fourier 2d filtrada
image(t(apply(log(abs(F_filtered) + 1), 2, rev)),
  col = gray.colors(256), axes = FALSE,
  main = "Transformada de Fourier 2d filtrada"
)
```



Ao aplicar o filtro passa-baixa e refazer o ffshift, podemos ver que as altas frequências foram zeradas, por condna disso a imagem da transformada de Fourier 2d filtrada mostra apenas as baixas frequências, que são os valores das bordas diferente do tom de cinze.

```
imGrayFiltered <- idf2d(F_filtered)

# Padronizando a imagem filtrada para [0, 1]
imGrayFiltered <- (imGrayFiltered - min(imGrayFiltered)) /
  (max(imGrayFiltered) - min(imGrayFiltered))
```

```
par(mfrow = c(1, 2))
image(
  t(imGrayG[nrow(imGrayG):1, ]),
  col = gray((0:255) / 255),
  main = "Imagem Original Cotaminada",
  axes = FALSE
)
image(
  t(imGrayFiltered[nrow(imGrayFiltered):1, ]),
  col = gray((0:255) / 255),
  main = "Imagem Suavizada",
  axes = FALSE
)
```

Imagem Original Cotaminada

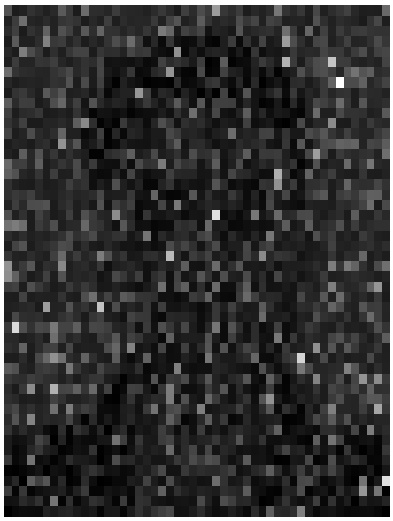
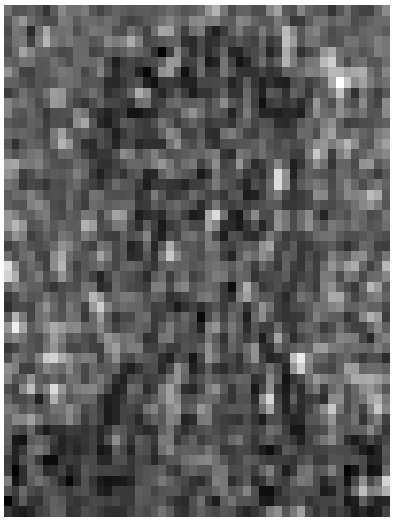


Imagem Suavizada



Depois de aplicar o filtro passa-baixa, podemos ver que a imagem apresentou uma melhora na visualização do contorno, apresentando um pouco do aspecto humano da imagem original, podendo ser percebido apenas com a visualização previa da imagem original.