

Exercício 2 - Template Matching

Rodrigo Machado Fonseca - 2017002253

October 31, 2021

1 Introdução

Neste trabalho utilizaremos o algoritmo *Template Matching* com o intuito de encontrar as placas 1 e 2 (figuras 1 e 2) na figura geral (figura 3).

2 Template Matching

O algoritmo Template Matching refere-se ao processamento de imagem em que encontramos modelos semelhantes em uma imagem de origem, fornecendo um modelo de base para comparação. O processo de correspondência do modelo é feito comparando cada um dos valores de pixel da imagem de origem, um de cada vez, com a imagem do modelo. A saída seria uma matriz de valores de similaridade em comparação com a imagem do modelo.

A comparação do algoritmo pode ser feita com uma métrica de distância. Neste estudo utilizaremos a métrica de Minkowsky:

$$\delta_{mink}(x_i, x_j) = \left(\sum_{k=1}^n |x_{ik} - x_{jk}|^p \right)^{\frac{1}{p}} \quad (1)$$

Na equação 1 podemos variar o valor de p e teremos valores diferentes para o mesmo conjunto de pontos. Para os valores de $p = 1$ e $p = 2$ a fórmula fica equivalente a distância de Manhattan e a distância Euclidiana.

3 Metodologia

Inicialmente, vamos carregar as imagens e transformá-las em escala de cinzas. O motivo disso é que, para figuras coloridas, cada pixel carrega três valores,

que representam os valores de RGB, enquanto que em escala de cinzas o pixel só carregará um único valor. Assim, ao passá-los para escala de cinzas, limitamos o nosso problema a um único cálculo de distância por pixel.



Figure 1: Placa 1



Figure 2: Placa 2

```
> rm(list=ls())
> library('jpeg')
> rotate <- function(x) t(apply(x, 2, rev))
> cor <- rev(gray(50:1/50))
> placa_1 <- readJPEG('placa_1.jpg', native = FALSE)
> image(rotate(placa_1[,,2]), col=cor)
> p1 <- rotate(placa_1[,,2])
> placa_2 <- readJPEG('placa_2.jpg', native = FALSE)
> image(rotate(placa_2[,,2]), col=cor)
> p2 <- rotate(placa_2[,,2])
> placas <- readJPEG('placas.jpg', native = FALSE)
> image(rotate(placas[,,2]), col=cor)
> Im <-rotate(placas[,,2])
```

Após a transformação em escala de cinza, para aplicarmos o Template Matching basta aplicarmos uma máscara do tamanho de cada placa no



Figure 3: Figura geral

conjunto com todas as placas. Essa máscara irá aplicar a distância de Minkowsky.

```
> minkowsky <- function(p, x1, x2){  
+   return((sum(abs(x1-x2)^p)^(1/p)))  
+ }  
> find_board <- function(p, p1, Im){  
+   nc <- (dim(Im)[2] - dim(p1)[2])  
+   nl <- (dim(Im)[1] - dim(p1)[1])  
+   d <- matrix(0, nl, nc)  
+   min_dist <- matrix(nrow = length(p), ncol = 1)  
+   d_list <- list()  
+   for(i in p){  
+     for(l in 1:nl){  
+       for(c in 1:nc){  
+         d[l,c] <- minkowsky(i, Im[l:(l+dim(p1)[1]-1),  
+                               c:(c+dim(p1)[2]-1)], p1)  
+       }  
+     }  
+     min_dist[i,1] <- min(d)  
+     d_list <- c(d_list, list(d))  
+   }  
+   return(list(min_dist, d_list))  
+ }
```

4 Resultados

Agora que possuímos nossa rotina de treino iremos avaliar p igual a 1, 2, 0.8, 4 para encontrar as placas 1 e 2. Abaixo mostraremos, para cada p , a superfície de contorno, o valor do ponto mínimo, o valor mínimo e a placa selecionada por um quadrado branco.

```
> p <- c(1, 2, 0.8, 4)
> results <- find_board(p, p1, Im)
> min_dist_1 <- results[1]
> d_list <- results[2]
> plot_image <- function(Im, result_p1, cor){
+   final1 <- Im
+   for (i in (result_p1[1]:(result_p1[1]+dim(p1)[1]-1)))
+   {
+     final1[i, result_p1[2]] = 1
+     final1[i, result_p1[2]+dim(p1)[2]-1] = 1
+     final1[i, result_p1[2]+1] = 1
+     final1[i, result_p1[2]+dim(p1)[2]] = 1
+     final1[i, result_p1[2]+2] = 1
+     final1[i, result_p1[2]+dim(p1)[2]+1] = 1
+   }
+   for (j in (result_p1[2]:(result_p1[2]+dim(p1)[2]-1)))
+   {
+     final1[result_p1[1],j] = 1
+     final1[result_p1[1]+dim(p1)[1]-1,j] = 1
+     final1[result_p1[1]+1,j] = 1
+     final1[result_p1[1]+dim(p1)[1],j] = 1
+     final1[result_p1[1]+2,j] = 1
+     final1[result_p1[1]+dim(p1)[1]+1,j] = 1
+   }
+   image(final1, col=cor)
+ }

> result_p1 <- which(d_list[[1]][[1]] == min(d_list[[1]][[1]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[1]])))

[1] "O valor mínimo é: " "99.8352941176471"
```

```
> contour(d_list[[1]][[1]])
```

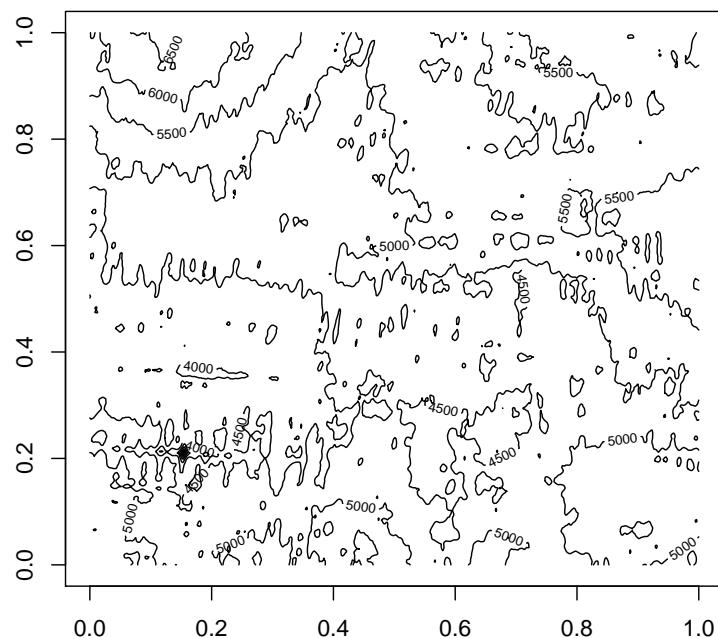


Figure 4: Superfície de contorno para placa 1 e p igual a 1.

```
> plot_image(Im, result_p1 = result_p1, cor = cor)
```



Figure 5: Imagem com a placa 1 selecionada no conjunto de placas, para p igual a 1.

```
> print(c("Está no ponto: ",
+         which(d_list[[1]][[1]] == min(d_list[[1]][[1]]), arr.ind = TRUE)))
[1] "Está no ponto: " "87"                      "71"

> print(c("Razão entre a média e o mínimo",
+         mean(d_list[[1]][[1]])/ min(d_list[[1]][[1]])))
[1] "Razão entre a média e o mínimo" "49.5964174587175"
```

```
> contour(d_list[[1]][[2]])
```

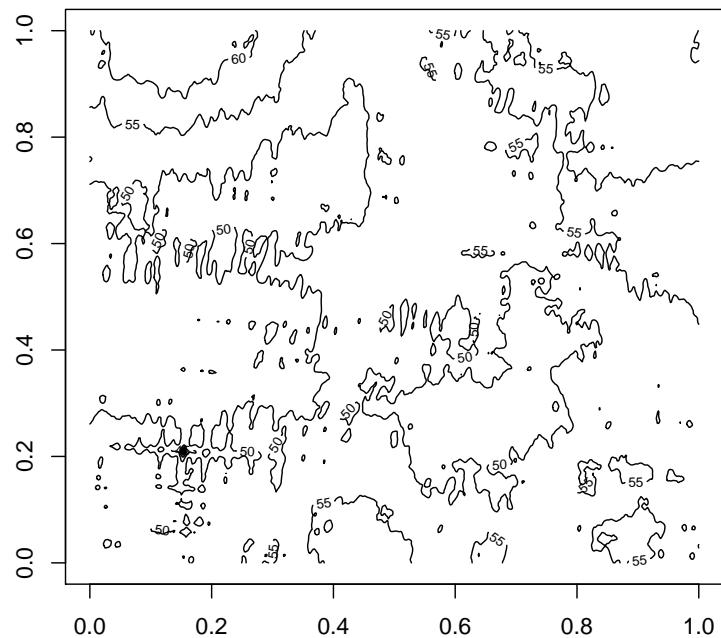


Figure 6: Superfície de contorno para placa 1 e p igual a 2.

```
> result_p2 <- which(d_list[[1]][[2]] == min(d_list[[1]][[2]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[2]])))

[1] "O valor mínimo é: " "1.1379984046952"

> print(c("Está no ponto: ",
+         which(d_list[[1]][[2]] == min(d_list[[1]][[2]]), arr.ind = TRUE)))

[1] "Está no ponto: " "87"           "71"

> print(c("Razão entre a média e o mínimo",
+         mean(d_list[[1]][[2]])/ min(d_list[[1]][[2]])))

[1] "Razão entre a média e o mínimo" "45.8000607641388"

> result_p08 <- which(d_list[[1]][[3]] == min(d_list[[1]][[3]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[3]])))

[1] "O valor mínimo é: " "985.613954530948"

> print(c("Está no ponto: ",
+         which(d_list[[1]][[3]] == min(d_list[[1]][[3]]), arr.ind = TRUE)))

[1] "Está no ponto: " "87"           "71"

> print(c("Razão entre a média e o mínimo",
+         mean(d_list[[1]][[3]])/ min(d_list[[1]][[3]])))

[1] "Razão entre a média e o mínimo" "51.1572353658518"

> result_p4 <- which(d_list[[1]][[4]] == min(d_list[[1]][[4]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[4]])))

[1] "O valor mínimo é: " "0.148135761890773"

> print(c("Está no ponto: ",
+         which(d_list[[1]][[4]] == min(d_list[[1]][[4]]), arr.ind = TRUE)))

[1] "Está no ponto: " "87"           "71"
```

```
> plot_image(Im, result_p1 = result_p2, cor = cor)
```



Figure 7: Imagem com a placa 1 selecionada no conjunto de placas, para p igual a 2.

```
> contour(d_list[[1]][[3]])
```

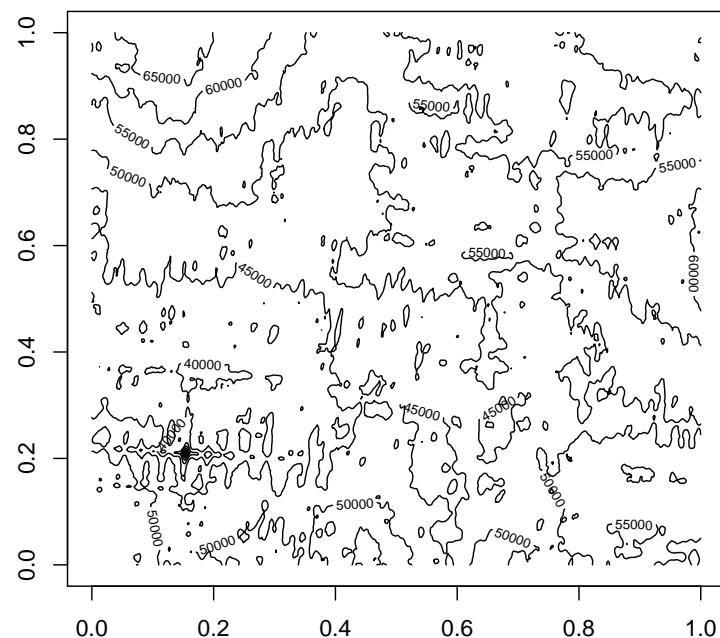


Figure 8: Superfície de contorno para placa 1 e p igual a 0.8.

```
> plot_image(Im, result_p1 = result_p08, cor = cor)
```



Figure 9: Imagem com a placa 1 selecionada no conjunto de placas, para p igual a 0.8

```
> contour(d_list[[1]][[4]])
```

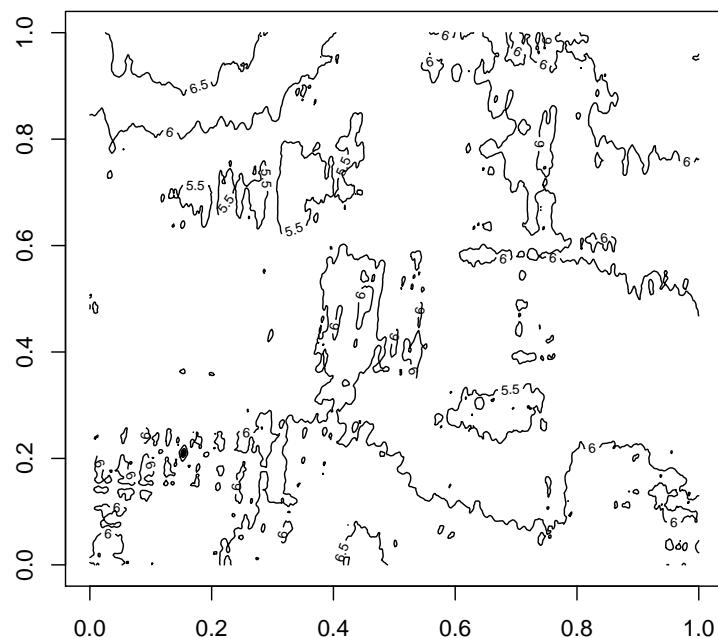


Figure 10: Superfície de contorno para placa 1 e p igual a 4.

```
> plot_image(Im, result_p1 = result_p4, cor = cor)
```



Figure 11: Imagem com a placa 1 selecionada no conjunto de placas, para p igual a 4.

```
> contour(d_list[[1]][[1]])
```

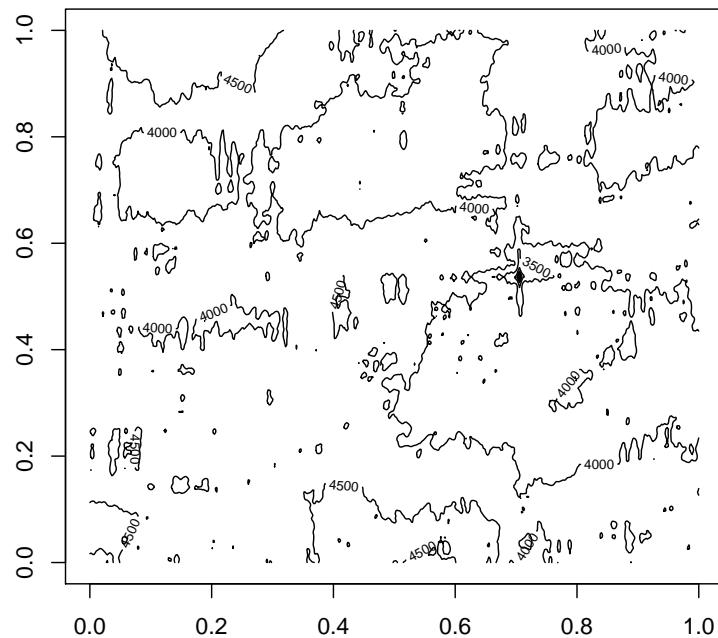


Figure 12: Superfície de contorno para placa 2 e p igual a 1.

```
> print(c("Razão entre a média e o mínimo",
+        mean(d_list[[1]][[4]])/ min(d_list[[1]][[4]])))

[1] "Razão entre a média e o mínimo" "39.8790442311116"

> p <- c(1, 2, 0.8, 4)
> results <- find_board(p, p2, Im)
> min_dist_1 <- results[1]
> d_list <- results[2]
```

```
> result_p1 <- which(d_list[[1]][[1]] == min(d_list[[1]][[1]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[1]])))

[1] "O valor mínimo é: " "117.76862745098"

> print(c("Está no ponto: ",
+         which(d_list[[1]][[1]] == min(d_list[[1]][[1]]), arr.ind = TRUE)))

[1] "Está no ponto: " "394"                      "183"

> print(c("Razão entre a média e o mínimo",
+         mean(d_list[[1]][[1]])/ min(d_list[[1]][[1]])))

[1] "Razão entre a média e o mínimo" "34.928060556941"

> result_p2 <- which(d_list[[1]][[2]] == min(d_list[[1]][[2]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[2]])))

[1] "O valor mínimo é: " "1.33881801577141"

> print(c("Está no ponto: ",
+         which(d_list[[1]][[2]] == min(d_list[[1]][[2]]), arr.ind = TRUE)))

[1] "Está no ponto: " "394"                      "183"

> print(c("Razão entre a média e o mínimo",
+         mean(d_list[[1]][[2]])/ min(d_list[[1]][[2]])))

[1] "Razão entre a média e o mínimo" "33.1503557828145"

> result_p08 <- which(d_list[[1]][[3]] == min(d_list[[1]][[3]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[3]])))

[1] "O valor mínimo é: " "1160.02884879164"

> print(c("Está no ponto: ",
+         which(d_list[[1]][[3]] == min(d_list[[1]][[3]]), arr.ind = TRUE)))

[1] "Está no ponto: " "394"                      "183"
```

```
> plot_image(Im, result_p1 = result_p1, cor = cor)
```



Figure 13: Imagem com a placa 2 selecionada no conjunto de placas, para p igual a 1.

```
> contour(d_list[[1]][[2]])
```

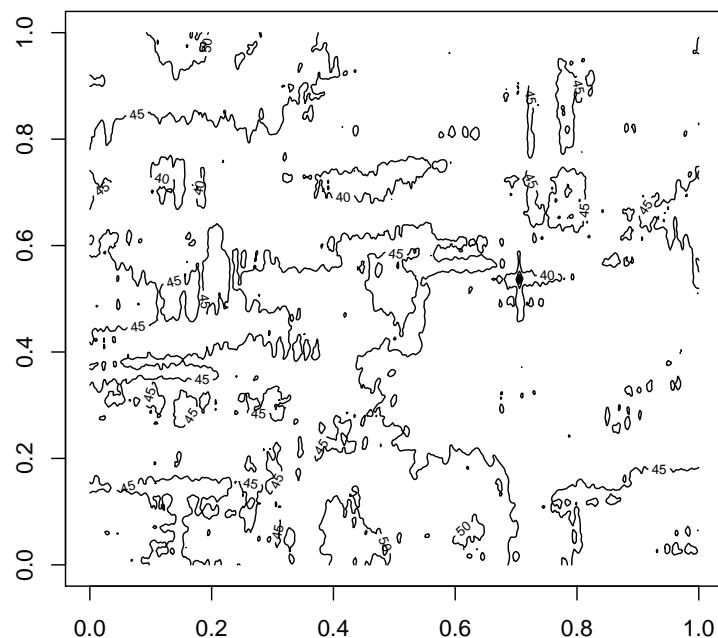


Figure 14: Superfície de contorno para placa 2 e p igual a 2.

```
> plot_image(Im, result_p1 = result_p2, cor = cor)
```



Figure 15: Imagem com a placa 2 selecionada no conjunto de placas, para p igual a 2.

```
> contour(d_list[[1]][[3]])
```

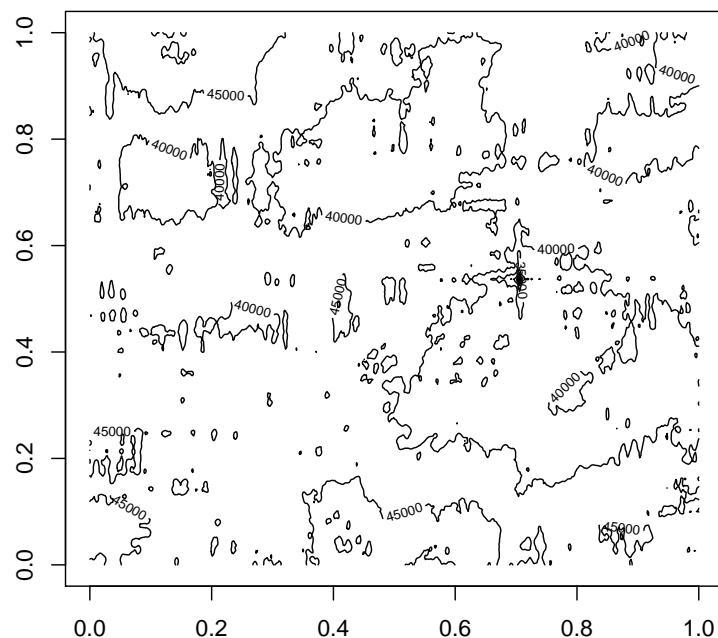


Figure 16: Superfície de contorno para placa 2 e p igual a 0.8.

```
> plot_image(Im, result_p1 = result_p08, cor = cor)
```



Figure 17: Imagem com a placa 2 selecionada no conjunto de placas, para p igual a 0.8

```
> contour(d_list[[1]][[4]])
```

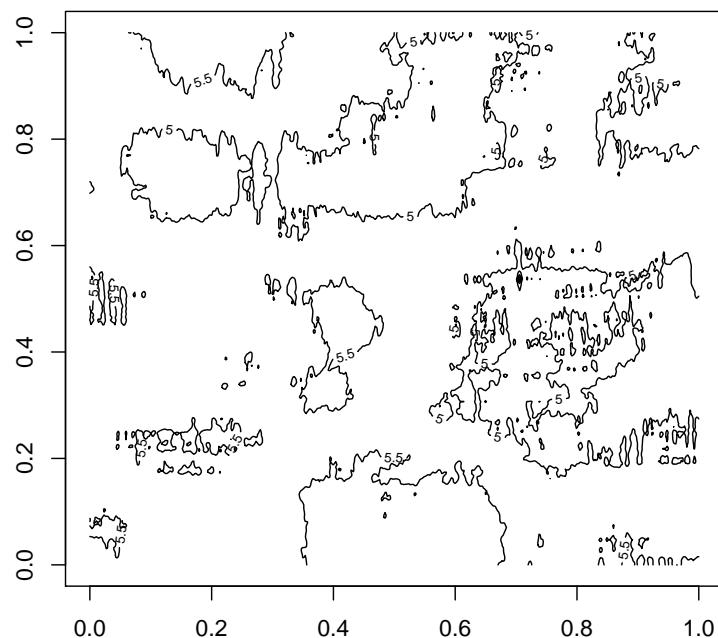


Figure 18: Superfície de contorno para placa 2 e p igual a 4.

```
> print(c("Razão entre a média e o mínimo",
+         mean(d_list[[1]][[3]])/ min(d_list[[1]][[3]])))

[1] "Razão entre a média e o mínimo" "35.7278894967374"

> result_p4 <- which(d_list[[1]][[4]] == min(d_list[[1]][[4]]), arr.ind = TRUE)
> print(c("O valor mínimo é: ", min(d_list[[1]][[4]])))

[1] "O valor mínimo é: " "0.168886345419138"

> print(c("Está no ponto: ",
+         which(d_list[[1]][[4]] == min(d_list[[1]][[4]]), arr.ind = TRUE)))

[1] "Está no ponto: " "394"                      "183"

> print(c("Razão entre a média e o mínimo",
+         mean(d_list[[1]][[4]])/ min(d_list[[1]][[4]])))

[1] "Razão entre a média e o mínimo" "30.7532181923793"
```

5 Discussão

A priori, pensei que o algoritmo que tivesse a menor distância seria o melhor. No entanto, quando queremos discernir algo, é melhor comparar o quanto os valores de distância se diferem uns dos outros ao longo da imagem. Isso é obtido ao fazer a razão entre a média e o mínimo. Dessa forma, o maior valor encontrado na razão corresponderá o melhor valor de p. Como podemos ver os melhores valores encontrados foram p igual 0.8 para as duas placas. Mas note que todos conseguiram de forma satisfatória encontrar a placa.

Outro fator importante é que quando analisamos os contornos para os melhores valores de p, figuras 8 e 16, podemos ver que há uma grande diferença entre o mínimo e as linhas de relevo mais próximas.

É valido comentar que o algoritmo se mostrou ineficiente em termos de tempo de execução. Por esse motivo, não foi possível expandir a análise para outros valores de p, já que levaria muito tempo para executar todo o código.

Pelos motivos supracitados, pode-se afirmar que os objetivos do experimento foram atingidos, uma vez que foi possível comparar métricas de distância e construir um algoritmo de Template Matching capaz de encontrar as duas placas de interesse.

```
> plot_image(Im, result_p1 = result_p4, cor = cor)
```



Figure 19: Imagem com a placa 2 selecionada no conjunto de placas, para p igual a 4.