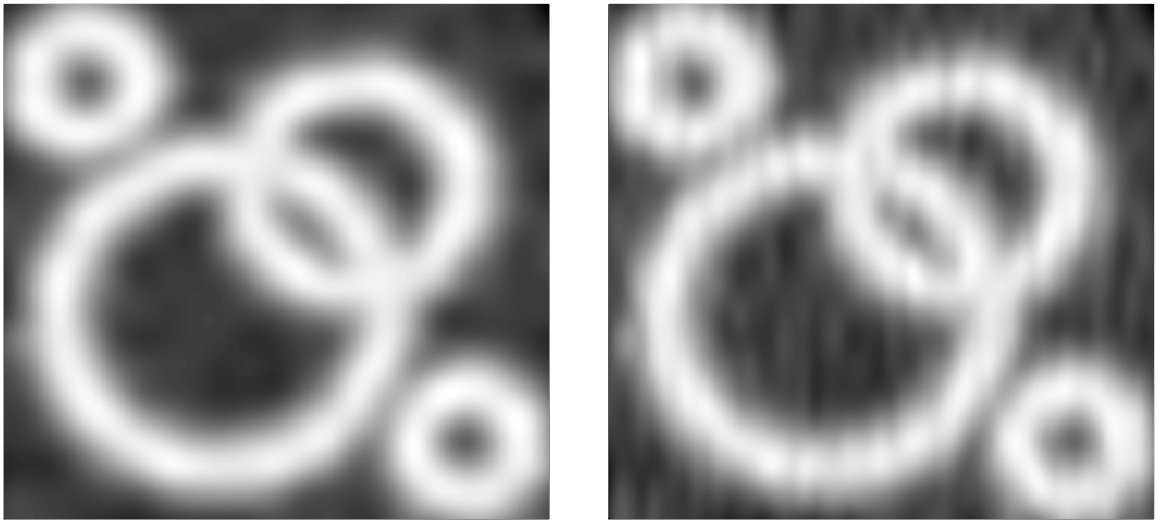
Effect of unequal segment numbers in 2D Bayesian smoothing

20 and 20 segments, $\lambda = 1.1$ | 50 and 20 segments, $\lambda = 2.1$



Bayesian isotropic smoothing of a noisy image (see Figure 4.10 for the data). Left: using bases with 20 segments. Right: one basis with 50, the other with 20 segments. The different sizes of the bases lead to artefacts, because the surface has much more freedom in the horizontal direction. R code in f-quasi-iso.R

```
# Effect of unequal segment numbers in 2D Bayesian smoothing (Simulated data)
# A graph in the book 'Practical Smoothing. The Joys of P-splines'
# Paul Eilers and Brian Marx, 2019

library(JOPS)
library(spam)

# Simulate the rings data
m1 = 500
m2 = 500
x1 = seq(-1, 1, length = m1)
x2 = seq(-1, 1, length = m2)
e1 = rep(1, m1)
e2 = rep(1, m2)
X1 = outer(x1, e2)
X2 = outer(e1, x2)
R1 = sqrt((X1 - 0.3)^2 + (X2 - 0.3)^2)
R2 = sqrt((X1 + 0.2)^2 + (X2 + 0.2)^2)
R3 = sqrt((X1 - 0.7)^2 + (X2 + 0.7)^2)
R4 = sqrt((X1 + 0.7)^2 + (X2 - 0.7)^2)
Y1 = exp(-50 * (R1 - 0.4)^2)
Y2 = exp(-50 * (R2 - 0.6)^2)
Y3 = exp(-50 * (R3 - 0.2)^2)
Y4 = exp(-50 * (R4 - 0.2)^2)
Y0 = pmax(pmax(pmax(Y1, Y2), Y3), Y4)
set.seed(123)
Y = Y0 + matrix(rnorm(m1 * m2), m1, m2)

cols = gray(seq(0, 1, by = 0.01))
par(mfrow = c(1, 2), mar = c(1, 1, 2, 1))

for (step in 1:2) {
  nseg1 = 50
```

```
nseg2 = 20
if (step ==1) nseg1 = 20

# Compute the inner product of the tensor product basis with array algorithm
B1 = bbase(x1, nseg = nseg1)
B2 = bbase(x2, nseg = nseg2)
n1 = ncol(B1)
n2 = ncol(B2)
W = 0 * Y + 1
T1 = rowtens(B1, B1)
T2 = rowtens(B2, B2);
BB = t(T1) %*% W %*% T2
dim(BB) = c(n1, n1, n2, n2)
BtB = aperm(BB, c(1, 3, 2, 4))
dim(BtB) = c(n1 * n2, n1 * n2)
Bty = as.vector(t(B1) %*% (W * Y) %*% B2)
yy = sum(Y ^ 2)

# Compute the penalty matrix (isotropic smoothing!)
d1 = d2 = 2
D1 = diff(diag(n1), diff = d1)
D2 = diff(diag(n2), diff = d2)
P1 = t(D1) %*% D1
P2 = t(D2) %*% D2
P = kronecker(P2, diag(n1)) + kronecker(diag(n2), P1)

# Makes matrices sparse
P = as.spam(P)
BtB = as.spam(BtB)

# Initialize
sig2 = 1;
tau2 = 1;
ndraw = 1000;
nomit = 100;
n = nrow(BtB)
As = matrix(0, ndraw, n);
ss = ts = d1s = d2s = rep(0, ndraw);

# Run the Markov chain
for (i in 1:ndraw) {
  if (i %% 100 == 1) cat('Step', i, '\n')

  # Update a
  Q = BtB / sig2 +  P / tau2;
  C = chol(Q);
  a0 = forwardsolve(C, Bty / sig2)
  a1 = backsolve(C, a0)
  a = a1 + backsolve(C, rnorm(n))
  As[i, ] = a

  # Update sigma^2
  ssq1 = yy - 2 * t(a) %*% Bty + t(a) %*% BtB %*% a
  d1s[i] = ssq1
  sig2  = c(ssq1 / rchisq(1, m1 * m2));
  ss[i] = sig2;

  #  Update tau^2
  ssq2 =  t(a) %*% P %*% a
  d2s[i] = ssq2;
  tau2  = c(ssq2 / rchisq(1, (n1 - d1) * (n2 - d2)));
  ts[i] = tau2;
}

# Compute average of coeffcients and the fit
amean = apply(As, 2, mean)
Amean = matrix(amean, n1, n2)
```

```
Z = B1 %*% Amean %*% t(B2)
lambda = mean(ss) / mean(ts)

# Prepare images/plots
image(x1, x2, Z, col = cols, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
titl = bquote('20 and 20 segments,' ~ lambda == .(round(lambda, 1)))
if(step!=1){
titl = bquote('50 and 20 segments,' ~ lambda == .(round(lambda, 1)))}
title(titl)
 }
```