

A Landmark-based Framework for Image Comparison in the Presence of Various Geometric Misalignments - Simulated Images

Anik Roy, Sagar Ghosh, Partha Sarathi Mukherjee

2026-03-28

```
#if (!requireNamespace("BiocManager", quietly = TRUE))
#  install.packages("BiocManager")

#BiocManager::install("EBImage")

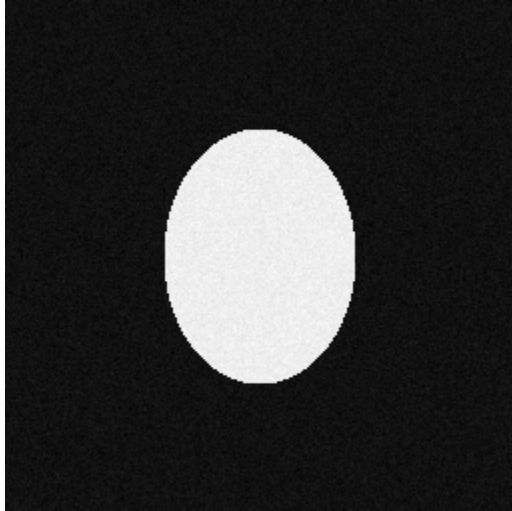
library(EBImage)
#library(raster)
library(ggplot2)
library(reshape2)
library(sp)
library(magick)
library(OpenImageR)
library(foreach)
library(doParallel)
library(DRIP)
library(twosamples)
library(OpenImageR)
library(magic)
library(pracma)
library(jpeg)
library(SAFARI)
#library(imager)
```

This file contains the R Codes to generate all the simulated images From Figure 4 in our paper.

Ellipse

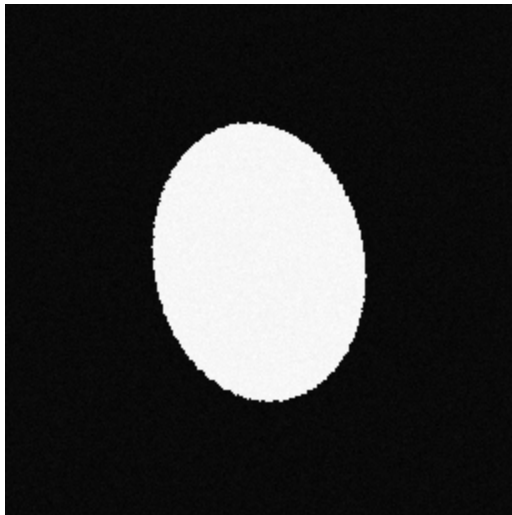
Null Ellipse $g_{E,0}$

```
x=matrix(0,256,256,dimnames=list(1:256,1:256))
for(i in 1:256){
  for(j in 1:256){
    if((((i-128)/48)^2+((j-128)/64)^2<1)){
      x[i,j]=0.5
    }
  }
}
nimg <- x + matrix(rnorm(nrow(x)*ncol(x),0,0.01),nrow=nrow(x),ncol=ncol(x))
image(nimg ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)
```



Null Rotated Ellipse $g_{E,1}^{(H_0)}$

```
x_1=matrix(0,256,256,dimnames=list(1:256,1:256))
for(i in 1:256){
  for(j in 1:256){
    if((((i-128)/48)^2+((j-128)/64)^2<1.2)){
      x_1[i,j]=1
    }
  }
}
x_1 <- rotateImage(x_1, 10, threads = 1)
nim1 <- x_1 + matrix(rnorm(nrow(x_1)*ncol(x_1),0,0.01),nrow=nrow(x_1),ncol=ncol(x_1))
image(nim1 ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)
```



Alternate Ellipse $g_{E,2}^{(H_1)}$

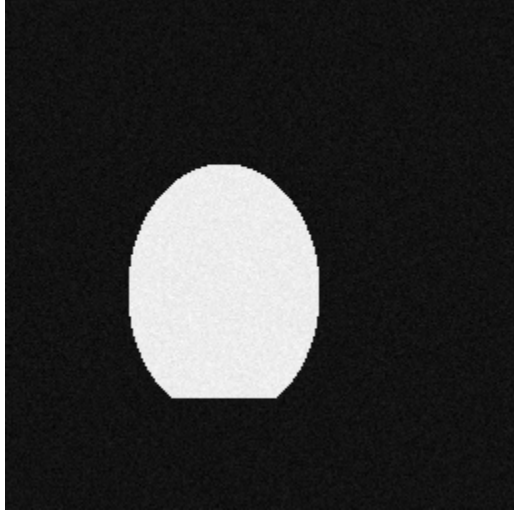
```
x_1=matrix(0,256,256,dimnames=list(1:256,1:256))
for(i in 1:256){
  for(j in 1:256){
    if((((i-110)/48)^2+((j-110)/64)^2<1)){
      x_1[i,j]=0.5
    }
  }
}
```

```

}

x_1[,1:56]=0
nimg1 <- x_1 + matrix(rnorm(nrow(x_1)*ncol(x_1),0,0.01),nrow=nrow(x_1),ncol=ncol(x_1))
image(nimg1 ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)

```



Alternate Ellipse $g_{E,3}^{(H_1)}$

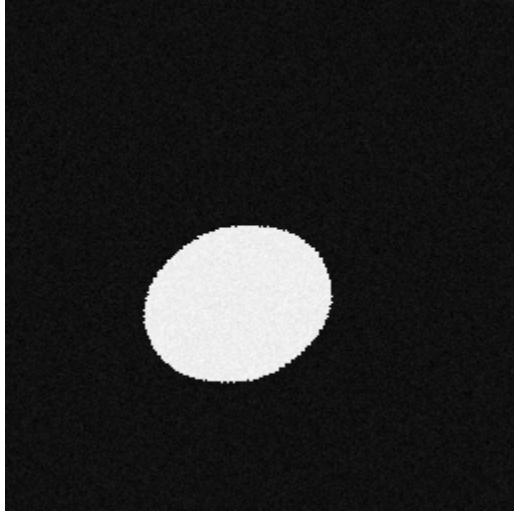
```

x_1=matrix(0,256,256,dimnames=list(1:256,1:256))
for(i in 1:256){
  for(j in 1:256){
    if((((i-110)/48)^2+((j-110)/38)^2<1)){
      x_1[i,j]=0.5
    }
  }
}

for(i in 1:256){
  for(j in 1:256){
    if((((i-110))^2+((j-110))^2<1500)){
      x_1[i,j]=0.5
    }
  }
}

x_1 <- rotateImage(x_1, 20, threads = 1)
nimg1 <- x_1 + matrix(rnorm(nrow(x_1)*ncol(x_1),0,0.01),nrow=nrow(x_1),ncol=ncol(x_1))
image(nimg1 ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)

```



Polygon

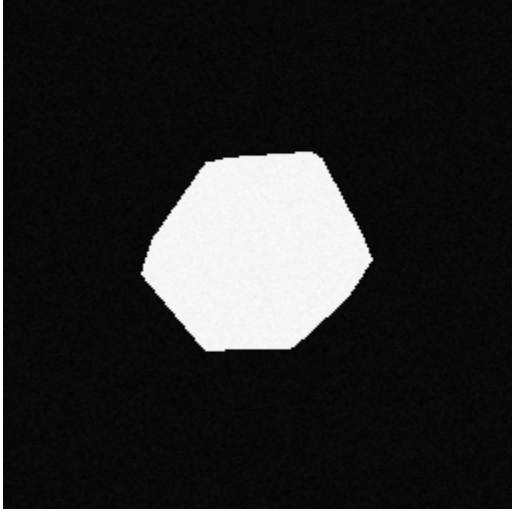
Null Polygon $g_{P,0}$

```
convex_polygon_image <- function(m=300, n=300, R=0.9, H=64) {
  set.seed(7)
  th <- runif(H, 0, 2*pi)
  # normals
  N <- cbind(cos(th), sin(th))
  # random radii around R (keep positive & bounded to remain convex and closed)
  r <- R * (0.7 + 0.3*runif(H))
  b <- r # since support in direction N_k is r_k

  xs <- seq(-1, 1, length.out = n)
  ys <- seq(-1, 1, length.out = m)
  XX <- matrix(rep(xs, each = m), nrow = m)
  YY <- matrix(rep(ys, times = n), nrow = m)

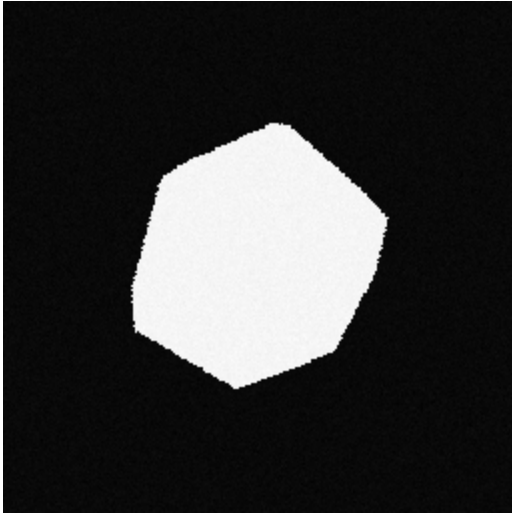
  img <- matrix(1L, m, n)
  for (k in seq_len(H)) {
    img[ (N[k,1]*XX + N[k,2]*YY) > b[k] ] <- 0L
  }
  img
}
y <- convex_polygon_image(256, 256, R=0.52, H=50)

nimgy <- y + matrix(rnorm(nrow(y)*ncol(y),0,0.01),nrow=nrow(y),ncol=ncol(y))
image(nimgy ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)
```



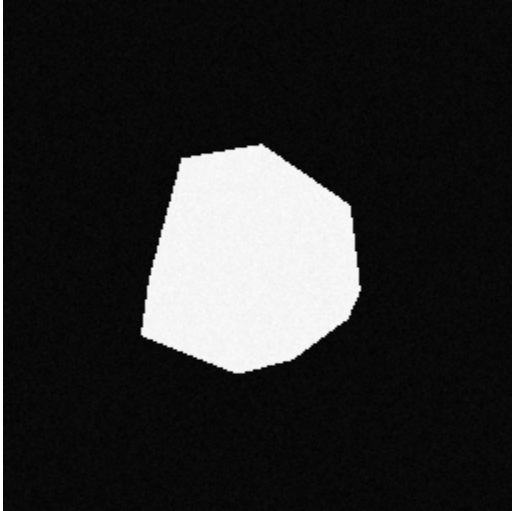
Null Polygon Rotated $g_{P,1}^{(H_0)}$

```
y_1 <- rotateImage(convex_polygon_image(256, 256, R=0.62, H=50),20 , threads = 1)
nimgy <- y_1 + matrix(rnorm(nrow(y_1)*ncol(y_1),0,0.01),nrow=nrow(y_1),ncol=ncol(y_1))
image(nimgy ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)
```



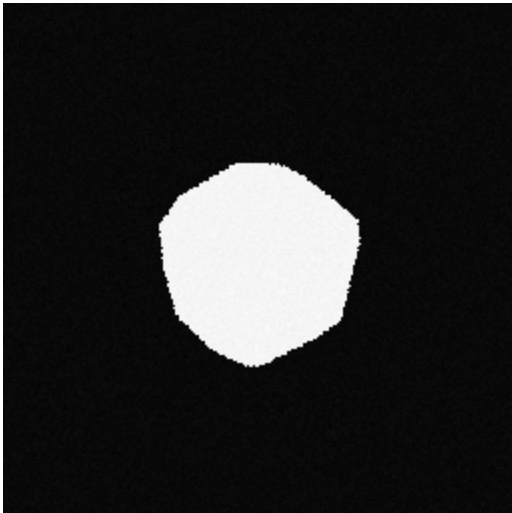
Alternate Polygon $g_{P,2}^{(H_1)}$

```
y_1 <- translation(convex_polygon_image(256, 256, R=0.52, H=20),shift_rows = 3,shift_cols = 2)
nimgy <- y_1 + matrix(rnorm(nrow(y_1)*ncol(y_1),0,0.01),nrow=nrow(y_1),ncol=ncol(y_1))
image(nimgy ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)
```



Alternate Polygon $g_{P,3}^{(H_1)}$

```
y_1 <- rotateImage(convex_polygon_image(256, 256, R=0.52, H=120), 20 , threads = 1)
nimgy <- y_1 + matrix(rnorm(nrow(y_1)*ncol(y_1),0,0.01),nrow=nrow(y_1),ncol=ncol(y_1))
image(nimgy ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)
```



Amoeba

Null Amoeba $g_{A,0}$

```
g= function(f,scale,mode=1)
{
  if(mode == 1) h= function(x) f(x)
  if(mode ==2) h= function(x) return(((f(x)*f(2*pi-x))^2)^0.25)
  img= array(1,dim=c(256,256))
  for(i in 1:256)
    for(j in 1:256)
    {
      loc1=i-128
      loc2=j-128
      theta = atan2(loc2, loc1)
```

```

theta = as.numeric(theta)[1]
if(theta < 0) theta = theta + 2*pi

#rat=loc2/loc1
#if(loc2==0) rat=0
#theta= atan(rat)
#if(loc2*theta < 0) theta = theta +pi
#if(length(theta) == 1 && theta == 0 && loc1 < 0) theta = pi

#if(theta ==0 & loc1 <0) theta =pi
#if(theta < 0) theta = 2*pi + theta
val = 1+h(theta)
dist = (loc1^2+loc2^2)^0.5
if(val*scale<dist)
  img[i,j]=0
}
return(img)
}
a=g(function(x) (sin(4*x)/3),50)
x=a
x_1 = rotateImage(x,20, threads = 1)
x_1= translation(g(function(x) (sin(4.5*x)/4),55), shift_rows = 3, shift_cols = 2)
x_1= rotateImage(g(function(x) (sin(4.35*x)/2.5),47), 20, threads = 1)

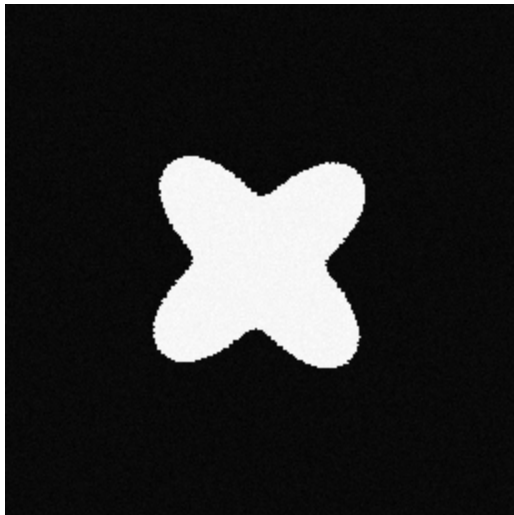
```

Null Rotated Amoeba $g_{A,1}^{(H_0)}$

```

x_1 = rotateImage(x,20, threads = 1)
nimgy <- x_1 + matrix(rnorm(nrow(x_1)*ncol(x_1),0,0.01),nrow=nrow(x_1),ncol=ncol(x_1))
image(nimgy ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)

```

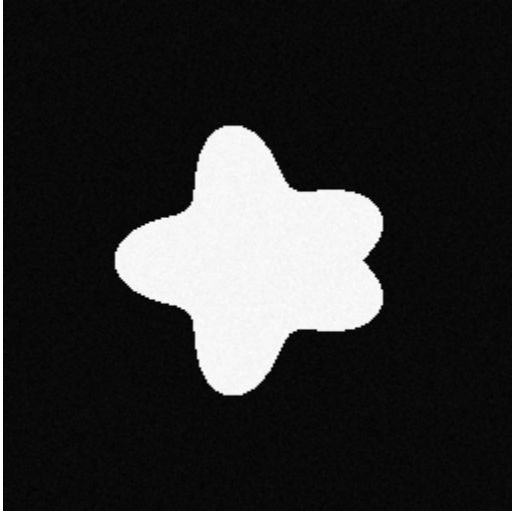


Alternate Amoeba $g_{A,2}^{(H_1)}$

```

x_1= translation(g(function(x) (sin(4.5*x)/4),55), shift_rows = 3, shift_cols = 2)
nimgy <- x_1 + matrix(rnorm(nrow(x_1)*ncol(x_1),0,0.01),nrow=nrow(x_1),ncol=ncol(x_1))
image(nimgy ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)

```



Alternate Amoeba $g_{A,3}^{(H_1)}$

```
x_1= rotateImage(g(function(x) (sin(4.35*x)/2.5),47), 20, threads = 1)
nimgy <- x_1 + matrix(rnorm(nrow(x_1)*ncol(x_1),0,0.01),nrow=nrow(x_1),ncol=ncol(x_1))
image(nimgy ,useRaster=TRUE,axes=FALSE,col = gray(0:256 / 256), asp=1)
```

