Package 'PET'

April 20, 2018

Version 0.5.1

Date 2018-04-20

Title Simulation and Reconstruction of PET Images

Author Joern Schulz <jschulz78@web.de>,
Peter Toft <pto@imm.dtu.dk>,
Jesper James Jensen <jjj@oedan.dk>,
Peter Philipsen <pap@imm.dtu.dk>

Maintainer Joern Schulz <jschulz78@web.de>

Depends R (>= 3.2.0), adimpro (>= 0.8.2), grDevices, graphics, stats,
utils

Description Implementation of different analytic/direct and
iterative reconstruction methods of radon transformed data
such as PET data. It also offer the possibility to simulate PET data.

License GPL (>= 2)

Copyright This package is Copyright (C) 2006-2018 by Joern Schulz and
Peter Toft.

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-20 12:50:10 UTC

R topics documented:

PET-package   *Simulation and Reconstruction of PET Images*

Description

This package implements different analytic or direct and iterative reconstruction methods of Peter
Toft. It also offer the possibility to simulate PET data.

Details

Package:	PET
Version:	0.4.8
Date:	2009-10-25
License:	GPL (>=2)
Copyright:	2006-2009 Joern Schulz and Peter Toft

Index:

```
cutMatrix           Cut matrix.
hough               Hough transformation.
iniPetFifList       Initializes a header for PET or FIF file.
iradonIT            Iterative inverse radon transformation.
iradon              Direct Inverse radon transformation.
markPoisson         Marked poisson process.
norm                L1 and L2 norm.
partEllipse         Generates a fragment of an ellipse.
phantom             Creation of a phantom.
radon               Radon transformation.
readData            Reload saved datasets in differnt formats.
readSifData         Reloaded system matrix.
rotate              Rotates data.
scaleImage          Scales an image.
viewData            Display several images.
writeData           Write datasets in different formats.
```

Author(s)

Joern Schulz <jschulz78@web.de>, Peter Toft <pto@imm.dtu.dk>, Jesper James Jensen <jjj@oedan.dk>, Peter Philipsen <pap@imm.dtu.dk>

Maintainer: Joern Schulz <jschulz78@web.de>

---

cutMatrix	*Cut Matrix*

Description

Scale the minimum of a matrix to 0 and cut off the first and the last rows or columns where the added up values are less than $1e - 10$.

Usage

```
cutMatrix(A, mode = "col")
```

Arguments

A	Describe the matrix.
mode	The default is set to mode="col" to cut off the first and the last columns of A where the added up columns values are less than $1e - 10$. Alternative choice is mode="row".

Value

A	The cut off matrix.
dimOrg	The dimension of the original image.
pattern	A pattern, contains the first and last column or row of A where the added up values are greater than $1e - 10$.

Author(s)

Joern Schulz <jschulz78@web.de>.

Examples

```
A <- matrix(c(rep(0,6),0:2,0,3,4,rep(0,4),5,6,rep(0,9)),nrow=3)
A
cutMatrix(A)
cutMatrix(A,mode="row")
rm(A)
```

---

hough | *Hough Transformation*

Description

The function implements a Hough transformation for an image.

Usage

```
hough(oData, mode=1, XYSamples=nrow(oData), DeltaXY=1.0,
      XYmin=-0.5*DeltaXY*(XYSamples-1), ThetaSamples=181,
      RhoSamples=2*round(sqrt(sum((dim(oData))^2))/2)+1, ThetaMin=0,
      RhoMin=-0.5*((2*round(sqrt(sum((dim(oData))^2))/2)+1)-1),
      DeltaTheta=pi/ThetaSamples, DeltaRho=(2*abs(RhoMin)+1)/RhoSamples,
      DebugLevel = "Normal")
```

Arguments

oData
: (matrix) A matrix that contains the image (for the Hough transformation).

mode
: (integer) The optimization strategy for the Hough transformation. Default is mode=1 for no optimization. Also implemented are mode=2 for 'vector mapping', mode=3 for 'avoid limit check and use vector mapping' and mode=4 for 'avoid limit check and use matrix mapping'.

XYSamples
: (integer) Specifies the number of samples on the x-axis (rows) and y-axis (columns) of oData. Defaults to XYSamples=nrow(oData).

DeltaXY
: (double) Specifies the sampling distance of both axes in the image. Defaults to DeltaXY=1.

XYmin
: (double) Specifies the minimum sample position in the image on the first and second axis. If not given, the image is centered around the middle. Defaults to XYmin=-0.5*DeltaXY*(XYSamples-1).

ThetaSamples
: (integer) Specifies the number of samples in the angular parameter θ in the Hough transformation image. It is sampled linearly from 0 to (approximately) π radians. Defaults to ThetaSamples=181.

RhoSamples
: Specifies the number of samples in the distance parameter ρ in the Hough transformation image.
  Defaults to RhoSamples=2*round(sqrt(sum((dim(oData))^2))/2)+1.

ThetaMin
: (double) Specifies the minimum sample position in the Hough transformation image on the first axis. Defaults to ThetaMin=0.

RhoMin
: (double) Specifies the minimum sample position in the Hough transformation image on the second axis.
  Defaults to RhoMin=-0.5*((2*round(sqrt(sum((dim(oData))^2))/2)+1)-1).

DeltaTheta
: (double) Angular sampling distance. Defaults to DeltaTheta=pi/ThetaSamples.

DeltaRho
: (double) Specifies the sampling distance in ρ. The program will center the sampling points around 0. Defaults to DeltaRho=(2*abs(RhoMin)+1)/RhoSamples

DebugLevel	(character) This parameter controls the level of output. Defaults to DebugLevel="Normal" for a standard level output. Alternative implementations are "Detail" if it is desirable to show almost all output on screen or "HardCore" for no information at all.

Details

It is shown in the paper of P.Toft that the Hough transformation can be defined in a way that gives exactly the same discrete parameter domain as found with the nearest neighbour approximation of the discrete Radon transformation. The Hough transformation does not have the same property. It is also shown that the Hough transformation behaves very differently when changing the sampling intervals in the discrete parameter domain, compared to the discrete Radon transformation. Furthermore different optimization schemes for the Hough transformation are described there.

Value

hData	A matrix, that contains the Hough transformation of oData.
Header	A list of following values:
	SignalDim The dimension of the hData.
	XYmin The minimum x- and y-position in hData.
	DeltaXY Sampling distance on the x- and y-axis in hData.
call	Arguments of the call to hough.

Author(s)

Peter Toft, Joern Schulz <jschulz78@web.de>.

References

Toft, Peter, *Ph.D. Thesis, The Radon Transform - Theory and Implementation*, Department of Mathematical Modelling Section for Digital Signal Processing, Technical University of Denmark, 1996. http://eivind.imm.dtu.dk/staff/ptoft/ptoft_papers.html

See Also

[radon](), [markPoisson]()

Examples

```
P <- phantom()
hP <- hough(P)
viewData(list(P, hP$hData), list("Phantom", "Hough transformed phantom"))
rm(P,hP)
```

---

iniPetFifList   *Initializes a Header for PET or FIF File*

Description

The function initializes a list with values that corresponds to a header of a ".pet" or a ".fif" file. The file structures of these formats are explicitly described in the functions readData and writeData of this package.

Usage

```
iniPetFifList(listType = "pet", data = NULL, imType = "normal")
```

Arguments

listType	Can be chosen between "pet" and "fif". Default is listType = "pet" to generate a default header for ".pet"-files. Choose listType = "fif" to generate a default header for ".fif"-files.
data	Using data, it is possible to consign the matrix corresponding to the list and to generate the parameters SignalDim, XYmin, DeltaXY and SignalMinMax with default values. Otherwise, if data = NULL these values will be set to NULL and you will have to specify these values.
imType	(character) imType specifies the type of image and thus the default values of the list. Default is imType="normal". It is also implemented imType="radon", i.e. that the image is a sinogram (radon transformed image).

Details

Set listType="pet" to generate a list with the following structure. It is assumed that data is a matrix and not equal to NULL.

Description	Contains a short description of the image. The maximal number of characters has to be 80. Defaults to Description="}\eqn{\backslash 0}{\0}\code{".
SignalDim	Contains the number of rows and columns of the matrix. Defaults to SignalDim=dim(data).
XYmin	Leftmost coordinate and lowest coordinate in the original image. If imType="normal" then defaults to XYmin= -0.5*(dim(data)-1) or if imType="radon" then defaults to XYmin=c(0, -0.5*(ncol(data)-1)).
DeltaXY	Quantization steps in the original image in x and y direction. If imType="normal" then defaults to DeltaXY=c(1,1) or if imType="radon" then defaults to DeltaXY= c(pi/(nrow(data)),1).

Set listType="fif" to generate a list, with the following structure. It is assumed that data is a

matrix and not equal to NULL.

FIFIdType	ID used to restore FIF:17737:'\0"\0"E"I'. Defaults to `FIFIdType=17737`.
FileName	Name used for saving/restoring this image. Defaults to `FileName="}\eqn{\backslash 0}{\0}\code{"`.
Description	See above.
Date	Date (YYYY-MM-DD). Defaults to `Date= as.character(Sys.Date())`.
SignalDim	See above.
ArrayType	Defines the format number. 1 for Real and 2 for Complex. Complex matrices are determined by $A = a_{i,j}$ where $a_{i,2n}$ is the imaginary part to the real value $a_{i,2n-1}, i = 1, \ldots, M, j = 1, \ldots, 2N, n = 1, \ldots, N$. Defaults to `ArrayType=1`.
XYmin	See above.
DeltaXY	See above.
SignalMinMax	The lowest and highest signal value in the matrix. Defaults to `SignalMinMax=c(min(data), max(data))`.

Value

Returns a list with above structure.

Author(s)

Joern Schulz <jschulz78@web.de>, Peter Toft.

See Also

writeData, readData

Examples

```
A <- phantom()
PetList <- iniPetFifList(data = A, imType = "normal")
PetList
rm(A,PetList)
```

---

iradon	*Direct Inverse Radon Transformation*

Description

The function implements different direct reconstruction methods for a radon transformed two-dimensional image.

Usage

```
iradon(rData, XSamples, YSamples, mode = "FB",
        Xmin = -sqrt(0.5)*(ncol(rData)/XSamples)*0.5*(XSamples-1),
        Ymin = -sqrt(0.5)*(ncol(rData)/YSamples)*0.5*(YSamples-1),
        DeltaX = sqrt(0.5)*(ncol(rData)/XSamples),
        DeltaY = sqrt(0.5)*(ncol(rData)/YSamples),
        InterPol=1, FilterTyp="Hamming1", oData=NULL,
        DebugLevel="Normal", iniFile=NULL)
```

Arguments

rData	(matrix) A matrix that contains the sinogram image (for the reconstruction functions).
XSamples	(integer) Number of samples on the x-axis (rows) in the reconstructed image.
YSamples	(integer) Number of samples on the y-axis (columns) in the reconstructed image.
mode	(character) The reconstruction method. Default is mode="FB" for Filtered Backprojection. Also implemented are "BF" for Filtering After Backprojection, "CNF" for Central Slice (FFT based with Nearest Neighbour approximation), "CBF" for Central Slice (FFT based with Bilinear Interpolation), "CC" for using a variation of the Central Slice (by the use of nonlinear sampling of the Radon domain with the use of the Chirp-z algorithm, the interpolation can be reduced to a simple one dimensional linear interpolation compared to normal CS), "CNC" for Central Slice (Chirp-z based with Nearest Neighbour approximation), "CBC" for Central Slice (Chirp-z based with Bilinear Interpolation) and "Test" to run all available reconstruction routines on the same image and to compare with a L1-Norm and L2-Norm. In case of "Test" the surrender value is NULL.
Xmin	(double) The minimum x-position of the reconstructed image. Defaults to Xmin=-sqrt(0.5)*(ncol(rData
Ymin	(double) The minimum y-position of the reconstructed image. Defaults to Ymin=-sqrt(0.5)*(ncol(rData
DeltaX	(double) Sampling distance on the x-axis. Defaults set to DeltaX = sqrt(0.5)*(ncol(rData)/XSamples).
DeltaY	(double) Sampling distance on the y-axis. Defaults set to DeltaY = sqrt(0.5)*(ncol(rData)/YSamples).
InterPol	(integer) Interpolation level. Used by Filtered Backprojection. Defaults InterPol=1.
FilterTyp	(character) Filter type is only used by Filtered Backprojection ("FB"). Defaults to FilterTyp="Hamming1" for generalized Hamming Filter with $\alpha = 0.5$. The alternative choice is "Hamming2" for generalized Hamming Filter with $\alpha = 0.54$ or a "Ramp" to get very good results of data without noise.
oData	(matrix) If mode="Test", measures of misfit can be made between rData and oData. oData is the original image of the radon transformed data. Defaults to oData=NULL.
DebugLevel	(character) This parameter controls the level of output. Defaults to DebugLevel="Normal" for a standard level output. Alternative implementations are "Detail" if it is desirable to show almost all output on screen or "HardCore" for no information at all.

iniFile   (character) If `iniFile!=NULL`, `iniFile` has to be the name of an ini-file including a pathname to the file. In the case of a specified `iniFile` all parameters are read from the file. Note that in this case contingently setting parameters (except for `DebugLevel`) in R are ignored when calling `iradon`. The parameters which are not specified in `iniFile` are set to defaults. Defaults to `iniFile=NULL`.

Details

The function implements different direct reconstruction methods for a radon transformed two-dimensional image. The different methods will be specified with parameter `mode`. The reconstruction methods are developed by P. Toft (1996) and implemented in R by J. Schulz (2006). For detail theoretical information about the radon-transformation see references.

Several things must be fulfilled to ensure a reasonable performance. Firstly sampling must be adequate in all parameters (see to references to get detail information). This will imply bounds on the sampling intervals. Secondly it is assumed that the fundamental function $f(x, y)$ to be reconstructed have compact support, or more precisely is zero if $\sqrt{x^2 + y^2} > |\rho_{max}|$. This demand will ensure that $Rf(\rho, \theta) = 0$, if $|\rho| > |\rho_{max}|$. If this cannot be fulfilled, numerical problems must be expected.

Assuming that $f(x, y)$ has compact support, then $(x, y) = (0, 0)$ should be placed to minimize $|\rho_{max}|$. This will reduce the size of the data array used for the discrete Radon transform.

Another variation to determine the parameter is the offer with `iniFile`. `iniFile` has to be a name of a file (e.g. `iniFile="/home/work/sino.ini"`) with the following structure. Each line begins with the name of parameter, then an equal sign follows and the value of the parameter. The first line must contain the parameter `mode`. Characters are not written in "", e.g. RadonKernel=NN and not RadonKernel="NN". Furthermore note that in an ini-file `rData` and `oData` both have to be of type character, videlicet the name of the corresponding file. Supported file formats are ".txt", ".dat", ".fif", ".pet", ".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif, ".jpg, ".jpeg. See ?readData to get detailed information about supported formats.

If a file of the type ".fif", ".pet" or ".dat" is specified for `oData` (see ?readData or ?writeData in R to get more information about these formats), the parameters XSamples, YSamples, XMin, YMin, DeltaX and DeltaY will be read from the file-header, but only in the case of unspecified corresponding parameters.

Note that in the case of an ini-file contingently setting parameters (except for `DebugLevel`) in R are ignored with the calling of `iradon`. Parameters that are not specified in the `iniFile` are set to default.

Value

irData   A matrix, that contains the reconstructed image (matrix) of `rData`.

Header   A list of following values:

                    SignalDim The dimension of the `irData`.

                    XYmin The minimum x- and y-position in `irData`.

                    DeltaXY Sampling distance on the x- and y-axis in `irData`.

call   Arguments of the call to `iradon`.

Author(s)

Joern Schulz <jschulz78@web.de>, Peter Toft.

References

Toft, Peter, *Ph.D. Thesis, The Radon Transform - Theory and Implementation*, Department of Mathematical Modelling Section for Digital Signal Processing, Technical University of Denmark, 1996. http://eivind.imm.dtu.dk/staff/ptoft/ptoft_papers.html

Schulz, Joern, *Diploma Thesis: Analyse von PET Daten unter Einsatz adaptiver Glaettungsverfahren*, Humboldt-Universitaet zu Berlin, Institut fuer Mathematik, 2006.

See Also

[radon](#), [iradonIT](#)

Examples

```
#
# Compare the results of different direct reconstruction methods
#
## Not run:
P <- phantom(design="B")
rP <- markPoisson(P, nSample=3000000 )
irP1 <- iradon(rP$rData , nrow(P), ncol(P))
irP2 <- iradon(rP$rData , nrow(P), ncol(P),
        mode="BF", DebugLevel="HardCore")
irP3 <- iradon(rP$rData , nrow(P), ncol(P),
        mode="CBF", DebugLevel="HardCore")
viewData(list(rP$rData, irP1$irData, irP2$irData, irP3$irData),
         list("Generated PET Data", "Reconstruction: mode='FB'",
         "Reconstruction: mode='BF'", "Reconstruction: mode='CBF'"))
rm(irP1,irP2,irP3,P,rP)

## End(Not run)

#
# Compare the results of different values for RhoSamples in 'markPoisson'
#
## Not run:
P <- phantom()
rP1 <- markPoisson(P, nSample=1000000, RhoSamples=101, image=FALSE)
rP2 <- markPoisson(P, nSample=1000000, RhoSamples=256, image=FALSE)
rP3 <- markPoisson(P, nSample=1000000, RhoSamples=501, image=FALSE)
rP4 <- markPoisson(P, nSample=1000000, RhoSamples=801, image=FALSE)
irP1 <- iradon(rP1$rData, 257, 257)
irP2 <- iradon(rP2$rData, 257, 257, DebugLevel="HardCore")
irP3 <- iradon(rP3$rData, 257, 257, DebugLevel="HardCore")
irP4 <- iradon(rP4$rData, 257, 257, DebugLevel="HardCore")
viewData(list(irP1$irData, irP2$irData, irP3$irData, irP4$irData,),
         list("RhoSamples=101",  "RhoSamples=256", "RhoSamples=501",
         "RhoSamples=801"))
```

```
        rm(P,rP1,rP2,rP3,rP4,irP1,irP2,irP3,irP4)

        ## End(Not run)

        #
        # mode="Test"
        #
        P <- phantom()
        R <- radon(P)
        iradon(R$rData, XSamples=257, YSamples=257, mode="Test", oData=P)
        rm(P,R)
```

---

iradonIT   *Iterative Inverse Radon Transformation*

Description

The function implements three of the major iterative reconstruction techniques: ART (Algebraic Reconstruction Technique), EM (Likelihood Reconstruction using Expectation Maximization) and LSCG (Least Squares Conjugate Method).

Usage

```
iradonIT(rData, XSamples, YSamples, StartImage = "None",
         mode = "EM", UseFast = 1, RadonKernel = "NN",
         Iterations = 20, IterationsType = "random",
         SaveIterations = 0, SaveIterationsName = "",
         LowestALevel = 0, ConstrainMin = -1,
         ConstrainMax = -1, Alpha = 1, Beta = 1,
         Regularization = 0, KernelFileSave = 0,
         KernelFileName = "", RefFileName = "None",
         ThetaSamples = nrow(rData), RhoSamples = ncol(rData),
         ThetaMin = 0, RhoMin = -0.5*((2*round(sqrt(XSamples^2+
         YSamples^2)/2)+1)-1), DeltaTheta = pi/ThetaSamples,
         DeltaRho = (2*abs(RhoMin)+1)/RhoSamples,
         Xmin = -0.5*(XSamples-1), Ymin = -0.5*(YSamples-1),
         DeltaX = 1, DeltaY = 1, OverSamp = 0,
         DebugLevel = "Normal", iniFile = NULL)
```

Arguments

rData	(matrix) A matrix that contains the sinogram. The rows of the image correspond to the sampled angles θ and the columns correspond to the sampled distance ρ. These both parameters determine the lines.
XSamples	(integer) XSamples is the number of samples on the x-axis (rows) in the reconstructed image.
YSamples	(integer) YSamples is the number of samples on the y-axis (columns) in the reconstructed image.

StartImage	(matrix) If provided, then the initial guess on the reconstructed image is taken from this image, otherwise a constant solution will be assumed from the start. Default is set to StartImage="None". Note, that the dimension of the StartImage has to be dim(StartImage)==c(XSamples,YSamples).
mode	(character) The iterative reconstruction method. Default is mode="EM" for Likelihood Reconstruction using Expectation Maximization. Additional implementations are "ART" (Algebraic Reconstruction Technique) and "CG" (Least Squares Conjugate Method).
UseFast	(logical) If UseFast=1 fast reconstruction is used, where the system matrix is stored (in case of KernelFileSave=1) using sparse techniques, otherwise a slower but more efficient memory reconstruction is used. Default is UseFast=1.
RadonKernel	(character) The type of kernel is used to model the system matrix. Default is RadonKernel="NN" for two-level Nearest Neighbour approximation (Memory consuming). Additional implementations are "RNN" for ray driven Nearest Neighbour approximation discrete Radon transformation based (very fast with small system matrix), "RL" for ray driven Linear Interpolation discrete Radon transformation based (fast with small system matrix), "P1" for method based on Radon transformation of square with pre-guidance (slow but good) and "P2" for method based on Radon transformation of square with no pre-guidance (slower but better).
Iterations	(integer) Using mode="EM" or "CG", it is the number of iterations before the iteration ends. Using mode="ART", it is the number of full iterations, i.e., divided by the number of rows in the system matrix. Defaults to Iterations=20.
IterationsType	(character) Using mode="ART", two iterations types are possible: "cyclic" selection of the row index or "random" selection. Defaults to IterationsType = "random".
SaveIterations	(integer) If it is set to SaveIterations=1, the current solution will be saved after each iteration under the name SaveIterationsName + CurrentIteration (also possible 2,3 ...). Defaults to SaveIterations=0. NOTE: When Iterations/itINI.SaveIterations>300 then too many new files are requested. In this case default is used.
SaveIterationsName	
	(character) In case of SaveIteration is non-zero, the parameter determines the name of the currently saved iteration. It contains the path to the file and the filename. The path contains the path relatively to the working-directory of your R-session or it contains the full path of the file. Defaults to SaveIterationsName="".
LowestALevel	(double) If fast reconstruction is used, the matrix elements will be truncated to this level relatively to the sampling distance of x. Defaults to LowestALevel=0.0.
ConstrainMin	(double) After each iteration the solution in each sample will be forced above this limit. Defaults to ConstrainMin=0.
ConstrainMax	(double) After each iteration the solution in each sample will be forced below this limit. Defaults to ConstrainMax=-1. NOTE: For both limit it is assumed that negative limits imply that the feature is not used.
Alpha	(double) If mode="ART", it will be the initial update weight. Defaults to Alpha=1.0.
Beta	(double) If mode="ART", it will be the multiplicative change to the weight factor, which should be less than one. Default is set to Beta=1.

Regularization	(double) If the Regularization is not zero and positive and using fast reconstruction, rows will be appended to the system matrix with a simple Laplace operator. A weight factor should also be incorporated. If negative, identity matrix will be used. Defaults to `Regularization=0`.
KernelFileSave	(logical) If `KernelFileSave=1` and fast reconstruction is used, the system matrix will be saved under the name `KernelFileName`. Defaults to `KernelFileSave=0`.
KernelFileName	(character) If fast reconstruction is used, the system matrix will be saved and restored with this sif-name. If the system matrix is incompatible to the sampling parameters, a new one will be generated. KernelFileName contains the path to the filename and the filename. The path contains the path relatively to the working-directory of your R-session or it contains the full path to the file. Defaults to `KernelFileName=""`.
RefFileName	(character) Error measures can be made between the image which is contained in `RefFileName` and the reconstructed image from `rData`. The file has to be either a fif-file or a pet-file (type `?writeData` or `?readData` in R to get more information about these formats). If `RefFileName` exists, the measures will be logged in a file with the name of `RefFileName` and extension 'dif'. Defaults to `RefFileName="None"`.
ThetaSamples	(integer) Number of angular samples θ in the sinogram. Defaults to `ThetaSamples=nrow(rData)`.
RhoSamples	(integer) Number of samples in the sinogram ρ in the ρ-direction. Defaults to `RhoSamples=ncol(rData)`.
ThetaMin	(double) Start of the angular sampling (should be set to zero). Defaults to `ThetaMin=0`.
RhoMin	(double) Start of sampling positions in ρ. Defaults to `RhoMin=-0.5*((2*round(sqrt(XSamples^2+YSamples^2)/2)+1)-1)`.
DeltaTheta	(double) Angular sampling distance. Should be set to the default `DeltaTheta = pi/ThetaSamples`.
DeltaRho	(double) Sampling distance in ρ. Defaults to `DeltaRho=(2*abs(RhoMin)+1)/RhoSamples`.
Xmin	(double) The minimum x-position of the reconstructed image. Defaults to `Xmin=-0.5*(XSamples-1)`.
Ymin	(double) The minimum y-position of the reconstructed image. Defaults to `Ymin=-0.5*(YSamples-1)`.
DeltaX	(double) Sampling distance on the x-axis. Defaults to `DeltaX=1`.
DeltaY	(double) Sampling distance on the y-axis. Defaults to `DeltaY=1`.
OverSamp	(integer) Uses oversampling - squared number of samples are used. Defaults to `OverSamp=0`.
DebugLevel	(character) This parameter controls the level of output. Defaults to `DebugLevel="Normal"` for a standard level output. Alternative implementations are `"Detail"` if it is desirable to show almost all output on screen or `"HardCore"` for no information at all.
iniFile	(character) If `iniFile!=NULL`, `iniFile` has to be the name of an ini-file including a pathname to the file. In the case of a specified `iniFile` all parameters are read from the file. Note that in this case contingently setting parameters (except for DebugLevel) in R are ignored when calling `iradonIT`. The parameters which are not specified in `iniFile` are set to defaults. Default of `iniFile` is `iniFile=NULL`.

Details

The function implements three of the major iterative reconstruction techniques: ART (Algebraic Reconstruction Technique), EM (Likelihood Reconstruction using Expectation Maximization) and LSCG (Least Squares Conjugate Method). The reconstruction methods were developed by P. Toft (1996) and implemented in R by J. Schulz (2006). For detailed theoretical information about the Radon-transformation see references.

There are a lot of different parameters which influence the reconstruction method. In common cases, it should be enough to determine rData, XSamples and YSamples. For the default parameters, it is assumed that the sinogram is standardised to a coordinate system with θ from 0 to π and ρ from $-0.5 * ((2 * round(sqrt(M^2 + N^2)/2) + 1) - 1)$ to $0.5 * ((2 * round(sqrt(M^2 + N^2)/2) + 1) - 1)$. It is also assumed that the reconstructed image is standardised to a coordinate system with x from $-0.5 * (M - 1)$ to $0.5 * (M - 1)$ and y from $-0.5 * (N - 1)$ to $0.5 * (N - 1)$. R, M and N are the number of sampling parameters in ρ, x and y. If that is not true, you will have to adapt the parameters.

Several things must be fulfilled to ensure a reasonable performance. Firstly sampling must be adequate in all parameters (see to references to get detail information). This will imply bounds on the sampling intervals. Secondly it is assumed that the fundamental function $f(x, y)$ to be reconstructed have compact support, or more precisely is zero if $\sqrt{x^2 + y^2} > |\rho_{max}|$. This demand will ensure that $Rf(\rho, \theta) = 0$, if $|\rho| > |\rho_{max}|$. If this cannot be fulfilled, numerical problems must be expected.

Assuming that $f(x, y)$ has compact support, then $(x, y) = (0, 0)$ should be placed to minimize $|\rho_{max}|$. This will reduce the size of the data array used for the discrete Radon transform.

The numerical complexity of the procedure is mainly determined by Iterations, the size of rData, XSamples, YSamples and by the use of fast reconstruction method.

Fast reconstruction is used in case of UseFast=1. It is possible to store the computed system matrix (KernelFileSave=1) under the name KernelFileName. The speed of the reconstruction can obviously be increased by using of an existing system matrix. NOTE: This is only possible, if the sampling parameters are compatible with the system matrix. That means, that the fifteen parameters XSamples, YSamples, DeltaX, DeltaY, Xmin, Ymin, ThetaSamples, RhoSamples, DeltaRho, DeltaTheta, ThetaMin, RhoMin, LowestALevel, RadonKernel and OverSamp are equal to the parameter from which the system matrix is generated. If the system matrix is incompatible to the sampling parameters, a new one will be generated.

Another variation to determine the parameter is the offer with iniFile. iniFile has to be the name of a file (e.g. iniFile="/home/work/sino.ini") with the following structure. Each line contains at first the name of a parameter, then an equal sign follows and the value of the parameter. The first line must contain the parameter mode. Characters are not written in "", e.g. RadonKernel=NN and not RadonKernel="NN". Furthermore note that in an ini-file rData and StartImage are to be of type character, videlicet the name of the corresponding file. Supported file formats are ".txt", ".dat", ".fif", ".pet", ".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg", ".jpeg". See ?readData to get detailed information about supported formats.

If a file of the type ".fif", ".pet" or ".dat" is specified for rData (see ?writeData or ?readData in R to get more information about these formats), the parameters ThetaSamples, RhoSamples, ThetaMin, RhoMin, DeltaTheta and DeltaRho will be read from the file-header of rData, but only in case of unspecified corresponding parameters. Just as, if a file of type ".fif", ".pet" or ".dat"

is specified for `StartImage`, then the parameters `XSamples`, `YSamples`, `XMin`, `YMin`, `DeltaX` and `DeltaY` will be read from the file-header.

Note that in case of an ini-file, contingently setting parameters (except for `DebugLevel`) in R are ignored when calling `iradonIT`. Parameters that are not specified in the `iniFile` are set to defaults.

Value

`irData`	A matrix, that contains the reconstructed image (matrix) of `rData`.
`Header`	A list of following values:
	SignalDim The dimension of the `irData`.
	XYmin The minimum x- and y-position in `irData`.
	DeltaXY Sampling distance on the x- and y-axis in `irData`.
`call`	Arguments of the call to `iradonIT`.

Author(s)

Joern Schulz <jschulz78@web.de>, Peter Toft.

References

Toft, Peter, *Ph.D. Thesis, The Radon Transform - Theory and Implementation*, Department of Mathematical Modelling Section for Digital Signal Processing, Technical University of Denmark, 1996. http://eivind.imm.dtu.dk/staff/ptoft/ptoft_papers.html

Schulz, Joern, *Diploma Thesis: Analyse von PET Daten unter Einsatz adaptiver Glaettungsverfahren*, Humboldt-Universitaet zu Berlin, Institut fuer Mathematik, 2006.

See Also

[radon](#), [iradon](#)

Examples

```
#
# Compare the results of iterative reconstruction method "EM" and
# direct reconstruction method "FB"
#
## Not run:
P <- phantom(design="B")
rP <- markPoisson(P, nSample=1600000 )
irP1 <- iradon(rP$rData , nrow(P), ncol(P))
irP2 <- iradonIT(rP$rData , nrow(P), ncol(P))
viewData(list(P, rP$rData, irP1$irData, irP2$irData),
         list("Generated unnoisy Phantom", "Generated PET Data",
         "Direct rec.: mode='FB'", "Iterative rec.: mode='EM'"))
rm(irP1,irP2,P,rP)

## End(Not run)

#
# Compare the results from the iterative reconstruction methods "EM"
```

```
# "CG" and "ART"
#
## Not run:
P <- phantom(n=151, addIm="blurred1")
rP <- markPoisson(P, nSample=2000000, RhoSamples=401)
irP1 <- iradonIT(rP$rData , nrow(P), ncol(P), Iterations=20,
        ConstrainMin=0, ConstrainMax=10)
irP2 <- iradonIT(rP$rData , nrow(P), ncol(P), mode="CG",
        Iterations=4, ConstrainMin=0, ConstrainMax=10)
irP3 <- iradonIT(rP$rData , nrow(P), ncol(P), mode="ART",
        Iterations=5, ConstrainMin=0, ConstrainMax=10,
        Alpha=0.1, Beta=0.5)
viewData(list(P,irP1$irData,irP2$irData,irP3$irData),
         list("Generated unnoisy Phantom",
         "mode='EM', Iterations=20","mode='CG', Iterations=4",
         "mode='ART', Iter.=20, Alpha=0.1, Beta=0.5"))
rm(irP1,irP2,irP3,P,rP)

## End(Not run)
```

---

markPoisson   *Marked Poisson Process*

Description

The function implements the Acceptance Rejection (AR) procedure to simulate a marked Poisson process with spatial varying intensity. Therewith it is possible to create noisy data from a phantom, as generated in Positron Emission Tomography (PET).

Usage

```
markPoisson(DataInt, nSample = 200000, ThetaSamples = 181,
            RhoSamples = 2*round(sqrt(sum((dim(DataInt))^2))/2)+1,
            RhoMin = -0.5*sqrt(2),
            vect.length = 100000, image = TRUE, DebugLevel = "Normal")
```

Arguments

DataInt   (matrix) A two dimensional image in which the matrix elements comply with intensities. That means the intensity of the decay of positrons in a certain tissue.

nSample   (integer) nSample determines the number of accepted events that will be generated with AR method. Defaults to nSample = 200000.

ThetaSamples (integer) Specifies the number of samples in the angular parameter θ in the sinogram. The sinogram is sampled linearly from 0 to (approximately) π radians. Defaults to ThetaSamples=181.

RhoSamples   (integer) Specifies the number of samples in the distance parameter ρ in the sinogram. Defaults to RhoSamples=2*round(sqrt(sum((dim(DataInt))^2))/2)+1.

RhoMin	(double) Specifies the minimum sample position in the sinogram on the second axis. It is assumed that `DataInt` is scaled to $[-0.5, 0.5]^2$. Defaults to `RhoMin=-0.5*sqrt(2)`.
vect.length	(integer) Determines a bound of number of generated accepted events in each iteration. That means, if `nSample > vect.length` then in each iteration `vect.length/(0.1217)` events will be generated and roughly `vect.length` accepted events. If you scale down `vect.length` the computation will be more time-consuming and if you increase the value of `vect.length`, more memory will be necessary. Defaults to `vect.length = 100000`.
image	(logical) If `image=FALSE`, only a generated sinogram will be returned, i.e. a noisy Radon transformation of the phantom will be returned. Otherwise, if `image=TRUE` an additional image will be returned, where each pixel of the image corresponds to a number of accepted events generated with the AR method. Defaults to `image=TRUE`.
DebugLevel	(character) This parameter controls the level of output. Available are: The default `DebugLevel="Normal"` for standard level output or alternative `"Detail"` if it desirable to logged almost all output to screen or `"HardCore"` for no information at all.

Details

The function implements the Acceptance Rejection (AR) method to simulate a marked Poisson process with spatial varying intensity. The function was developed to simulate data of a PET scanner. Therefore, new random events are generated with the AR method from `DataInt`. An angle θ will be generated to each event (uniform and iid. in $[0, \pi]$), because a PET scanner detects two photons, who will travel in (nearly) opposite directions. The line between the two detectors has the line parameters (ρ, θ), whereas ρ is the shortest distance from the origin of the coordinate system to the line, and θ an angle corresponding to the angular orientation of the line. The only obtainable information from the two photons is the fact, that the photon emission took place somewhere along that line.

Value

rData	A matrix, that contains the Radon transformed data.
Data	Will be returned only in case of `image = TRUE`. A matrix, where each pixel of the image corresponds to a number of accepted events generated with the AR method.
Header	A list of following values:
	SignalDim The dimension of the `rData`.
	XYmin The minimum x- and y-position in `rData`.
	DeltaXY Sampling distance on the x- and y-axis in `rData`.
call	Arguments of the call to `markPoisson`.

Author(s)

Joern Schulz <jschulz78@web.de>.

References

Schulz, Joern, *Diploma Thesis: Analyse von PET Daten unter Einsatz adaptiver Glaettungsverfahren*, Humboldt-Universitaet zu Berlin, Institut fuer Mathematik, 2006.

Gentle, J.E., *Elements of Computational Statistics*, Springer-Verlag New York Berlin Heidelberg, 2002.

See Also

radon, iradon, iradonIT

Examples

```
## Not run:
P <- phantom()
rP <- radon(P)
mP1 <- markPoisson(P)
mP2 <- markPoisson(P, nSample = 1000000)
viewData(list(P, mP1$Data, mP2$Data, rP$rData, mP1$rData, mP2$rData),
         list("Phantom", "nSample = 200000", "nSample = 1000000",
              "Radon Transfom of Phantom",  "nSample = 200000",
              "nSample = 1000000"))
cat("Number of generated accepted events for mP2:",sum(mP2$Data),"\n")
rm(mP1,mP2,P,rP)

## End(Not run)
```

---

norm   *L1 and L2 norm*

Description

Computes the L1 or L2 norm between two vectors or matrices.

Usage

```
norm(orgImage, testImage, mode = "L2")
```

Arguments

orgImage	A vector or a matrix.
testImage	A vector or a matrix of same size as orgImage.
mode	Defaults to mode="L2" to compute a L2 norm between orgImage and testImage. Alternative choice is "L1".

Details

If X and Y are vectors of length n or (n_1, n_2)-matrices with $n = n_1 * n_2$ then the L1-norm and the L2-norm will be compute as follow:

$$L1 = \frac{1}{n} \sum_{i=1}^{n} |X_i - Y_i|$$

$$L2 = \left(\frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i)^2 \right)^{\frac{1}{2}}$$

Value

Returns the L1 or L2 norm.

Author(s)

Joern Schulz, <jschulz78@web.de>.

Examples

```
P <- phantom(n=101)
P.pois <- markPoisson(P)$Data
cat("The L1 is:",norm(P, P.pois, mode="L1"),"\n")
rm(P,P.pois)
```

---

partEllipse   *Generates a Fragment of an Ellipse*

Description

The function offers the possibility to generate and to manipulate an ellipse. The data will be generated in $[-1, 1]^2$.

Usage

```
partEllipse(mod = "hcirc1", x = 0, y = 0, intensity = 0.1,
            n = 257, re1 = 0.2, re2 = 0.2, ring.wide = 0.05,
            no.xax1 = -1, no.xax2 = 1, no.yax1 = -1, no.yax2 = 1,
            in.r = 0.01, DebugLevel = "HardCore")
```

Arguments

mod	Defines the type of computation technique. Default is mod = "hcirc1", i.e. the ellipse is generated only by the parameters from mod to no.yax2. Also implemented are mod = "hcirc2" and mod = "hcirc3" where additionally in each point a circle with radius in.r will be generated in the area with an intensity. The effect is that the border of the ellipse is rounded. If mod = "hcirc2" all values in an accepted area will be set to intensity, but if mod = "hcirc3" then values in an accepted area will be cumulated. The result is more blurred. Note that in both cases the costs for computation increase obviously, all the more value exist with intensity!=0.
x	Is the x-coordinate of the center of the ellipse. Defaults to x = 0.
y	Is the y-coordinate of the center of the ellipse. Defaults to y = 0.
intensity	The intensity of the values, those are in the defined area. Defaults to intensity = 0.1.
n	(integer) Is the number of columns and rows in the generated image. It is assumed that the number of columns are equal to the number of rows. Defaults to n = 257.
re1	Is the half length of the horizontal axis. Defaults to re1 = 0.2.
re2	Is the half length of the vertical axis. Defaults to re2 = 0.2.
ring.wide	Defines the wide of the ellipse. This area is set to intensity. Note $0 <$ ring.wide $<= 1$. Defaults to ring.wide = 0.05.
no.xax1	The area of x-axis between -1 and no.xax1 is set to 0. Defaults to no.xax1 = -1.
no.xax2	The area of x-axis between 1 and no.xax2 is set to 0. Defaults to no.xax2 = 1.
no.yax1	The area of y-axis between -1 and no.yax1 is set to 0. Defaults to no.yax1 = -1.
no.yax2	The area of y-axis between 1 and no.yax2 is set to 0. Defaults to no.yax2 = 1.
in.r	If mod = "hcirc2" or mod = "hcirc3" in.r will define the radius of circles that will be generated in each point of the areas with itensity > 0. If mod="hcirc1" in.r will be ignored. Defaults to in.r = 0.01.
DebugLevel	(character) This parameter controls the level of output. Defaults to DebugLevel="HardCore" for no information at all. Alternative implementations are "Detail" if it is desirable to show almost all output on screen or "Normal" for a standard level output.

Value

Returns a matrix.

Author(s)

Joern Schulz <jschulz78@web.de>.

See Also

phantom

Examples

```
## Not run:
ellip1 <- partEllipse()
ellip2 <- partEllipse(x=0.1, y=-0.1, ring.wide=0.4)
ellip3 <- partEllipse(mod="hcirc2", y=0.2, intensity=0.4,
                      re1=0.5, no.xax1=0, DebugLevel="Normal")
ellip4 <- partEllipse(mod="hcirc3", y=0.2, intensity=0.4,
                      re1=0.5, no.xax2=0, DebugLevel="Normal")
viewData(list(ellip1, ellip2, ellip3, ellip4,),
         list("Part of an ellipse 1", "Part of an ellipse 2",
              "Part of an ellipse 3", "Part of an ellipse 4"))
rm(ellip1,ellip2,ellip3,ellip4)

## End(Not run)
```

---

phantom   *Creation of a Phantom*

Description

This function creates two dimensional phantom data. There are different options, e.g. to generate a big ellipse that represents the head, and several smaller ellipses, that represent pathological areas to be located in the space of the bigger ellipse.

Usage

```
phantom(n = 257, design = "A", addIm = "none", DebugLevel = "Normal")
```

Arguments

n	(integer) Is the number of columns and rows in the generated phantom. It is assumed that the number of columns is equal to the number of rows. Defaults to n=257.
design	(character) design characterizes the phantom data. It is possible to define different ellipses, with different intensities. There are four default-designs, these are design = "A", "B", "C" and "D". To define an own design of the phantom see below to details. Defaults to design = "A".
addIm	(character) Adds an additional image to the phantom. There are six default-designs for addIm. These are addIm="blurred1", "blurred2", "keen1", "keen2", "simple1" and "simple2". The defaults of addIm will be generated with the function partEllipse. If addIm="none", no image will be added to the phantom. A further possibility is a matrix A (addIm=A) with the same size as the phantom (i.e. dim(A)==c(n,n)). Defaults to addIm = "none".
DebugLevel	(character) This parameter controls the level of output. Defaults to DebugLevel="Normal" for a standard level output. Alternative implementations are "Detail" if it is desirable to show almost all output on screen or "HardCore" for no information at all.

Details

`design:`

To define an own design of the phantom (e.g. `design=P`) you have to note the following conditions:

1. `P` is a $(n, 5)$ or $(n, 6)$ matrix, whereas $n > 0$.

2. All elements x of `P` have to be between -1 and 1.

3. Each row from `P` define a ellipse on $[-1, 1]^2$, where

[,1]	A	Is the additive intensity of the corresponding ellipse.
[,2]	x_0	Is the x-coordinate of the center of the ellipse.
[,3]	y_0	Is the y-coordinate of the center of the ellipse.
[,4]	a	Is the half length of the horizontal axis.
[,5]	b	Is the half length of the vertical axis.
[,6]	α	Is the angle in degree between the x-axis of the ellipse and the x-axis of the grid. This parameter is optional, but the number of columns have to be the same in all rows.

Value

Returns a phantom image.

Author(s)

Joern Schulz <jschulz78@web.de>.

See Also

[partEllipse](), [markPoisson]()

Examples

```
P1 <- phantom()
P2 <- phantom(addIm="blurred1")
PhPa1 <- c(0.5, 0, 0, 0.4, 0.6)
PhPa2 <- matrix( c(0.6, -0.35, 0, 0.4, 0.6,
        0.3,  0.5,  0, 0.2, 0.35), nrow=2, byrow=TRUE )
P3 <- phantom(design=PhPa1)
P4 <- phantom(design=PhPa2)
viewData(list(P1, P2, P3, P4), list("Default Phantom",
        "addIm='blurred1'", "First new design",
        "Second new design"))
rm(P1,P2,P3,P4,PhPa1,PhPa2)
```

---

radon   *Radon Transformation*

Description

This function will forward project (normal Radon transformation) a quadratic image into a sinogram.

Usage

```
radon(oData, mode="NN", XYSamples=nrow(oData),
      XYmin=-0.5*(nrow(oData)-1), DeltaXY=1, ThetaSamples=181,
      RhoSamples=2*round(sqrt(sum((dim(oData))^2))/2)+1, ThetaMin=0,
      RhoMin=-0.5*((2*round(sqrt(sum((dim(oData))^2))/2)+1)-1),
      DeltaTheta=pi/ThetaSamples, DeltaRho=(2*abs(RhoMin)+1)/RhoSamples)
```

Arguments

oData	(matrix) A matrix that contains the image (for the Radon transformation).
mode	(character) The interpolation procedure. Currently supported functions are mode="NN" (Nearest Neighbour Interpolation), "LI" (Linear Interpolation) and "SINC" (Sinc Interpolation). Defaults to mode="NN".
XYSamples	(integer) Specifies the number of samples on the x-axis (rows) and y-axis (columns) in oData. Defaults to XYSamples=nrow(oData).
XYmin	(double) Specifies the minimum sample position in the image on the first and second axis. If not given, the image is centered around the middle. Defaults to XYmin=-0.5*(nrow(oData)-1).
DeltaXY	(double) Specifies the sampling distance of both axes in the image. Defaults to DeltaXY=1.
ThetaSamples	(integer) Specifies the number of samples in the angular parameter θ in the sinogram. The sinogram is sampled linearly from 0 to (approximately) π radians. Defaults to ThetaSamples=181.
RhoSamples	(integer) Specifies the number of samples in the distance parameter ρ in the sinogram. Should be an odd number. Defaults to RhoSamples = 2*round(sqrt(sum((dim(oData))^2))/2)+1.
ThetaMin	(double) Specifies the minimum sample position in the sinogram on the first axis. Defaults to ThetaMin=0.
RhoMin	(double) Specifies the minimum sample position in the sinogram on the second axis. Defaults to RhoMin=-0.5*((2*round(sqrt(sum((dim(oData))^2))/2)+1)-1).
DeltaTheta	(double) Angular sampling distance. Defaults to DeltaTheta=pi/ThetaSamples.
DeltaRho	(double) Specifies the sampling distance in ρ. The program will center the sampling points around 0. Defaults to DeltaRho=(2*abs(RhoMin)+1)/RhoSamples.

Details

The function implements a forward projection of a quadratic image. The used projection is the Radon transformation, named after Johann Radon (1917). The normal form of a line is given by

$$\rho = x\cos(\theta) + y\sin(\theta).$$

There are different ways to define a Radon transformation of a two dimensional continuous function f. One possibility is to integrate values of f along lines, whereas the location of the line is determined by the parameter ρ, the shortest distance from the origin of the coordinate system to the line, and θ an angle corresponding to the angular orientation of the line. If negative values of ρ are introduced the parameter domain is bounded by $0 \le \theta < \pi$ and $-\rho_{max} \le \rho \le \rho_{max}$. Numerical computations use the discrete Radon transformation. Therefore all parameters are sampled linearly

$$x = x_m = XMin + m * DeltaX, m = 0, \ldots, XSamples - 1,$$
$$y = y_n = YMin + n * DeltaY, n = 0, \ldots, YSamples - 1,$$
$$\rho = \rho_r = RhoMin + r * DeltaRho, r = 0, \ldots, RhoSamples - 1 \text{ and}$$
$$\theta = \theta_t = ThetaMin + t * DeltaTheta, t = 0, \ldots, ThetaSamples - 1.$$

Given the digitally sampled image a fundamental problem arises. The discrete Radon transformation requires samples that are not found in the digital image, because linear sampling of all variables implies that sampling parameter in the Radon transformation in general never coincides with the sampling parameter of the digital image. That should be clear and it is necessary to approximate the x and y by using `"NN"` (nearest neighbour approximation), `"LI"` (linear interpolation) or `"SINC"` (sinc interpolation).

Several things must be fulfilled to ensure a reasonable performance. Firstly sampling must be adequate in all parameters (see to references to get detail information). This will imply bounds on the sampling intervals. Secondly it is assumed that the fundamental function $f(x, y)$ to be reconstructed have compact support, or more precisely is zero if $\sqrt{x^2 + y^2} > |\rho_{max}|$. This demand will ensure that $Rf(\rho, \theta) = 0$, if $|\rho| > |\rho_{max}|$. If this cannot be fulfilled, numerical problems must be expected.

Assuming that $f(x, y)$ has compact support, then $(x, y) = (0, 0)$ should be placed to minimize $|\rho_{max}|$. This will reduce the size of the data array used for the discrete Radon transform.

The forward projection methods that are implemented in this function are developed by P. Toft (1996) and implemented in R by J. Schulz (2006). For example see to references to get more detailed theoretical information about the Radon transformation.

Value

rData   A matrix, that contains the Radon transformed data.

Header   A list of following values:

        SignalDim The dimension of the rData.

        XYmin The minimum x- and y-position in rData.

        DeltaXY Sampling distance on the x- and y-axis in rData.

call   Arguments of the call to radon.

Author(s)

Joern Schulz `<jschulz78@web.de>`, Peter Toft.

References

Toft, Peter, *Ph.D. Thesis, The Radon Transform - Theory and Implementation*, Department of Mathematical Modelling Section for Digital Signal Processing, Technical University of Denmark, 1996. http://eivind.imm.dtu.dk/staff/ptoft/ptoft_papers.html

Schulz, Joern, *Diploma Thesis: Analyse von PET Daten unter Einsatz adaptiver Glaettungsverfahren*, Humboldt-Universitaet zu Berlin, Institut fuer Mathematik, 2006.

Radon, Johann, *Ueber die Bestimmung von Funktionen durch ihre Integralwerte laengs gewisser Mannigfaltigkeiten*, Berichte ueber die Verhandlungen der koeniglich saechsischen Gesellschaft der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse, Band 69, 1917.

See Also

[markPoisson](), [iradon](), [iradonIT]()

Examples

```
P <- phantom()
R1 <- radon(P)
R2 <- radon(P, DeltaRho=0.5)
R3 <- radon(P, RhoSamples=501)
R4 <- radon(P, RhoSamples=251, RhoMin=-125, DeltaRho=1)
R5 <- radon(P, RhoSamples=451, RhoMin=-225, DeltaRho=1)
viewData(list(P, R1$rData, R2$rData, R3$rData, R4$rData, R5$rData),
         list("Phantom P with 256x256 voxels", "Default sinogram of P",
              "DeltaRho=0.5", "RhoSamples=501",
              "RhoSamples=251, RhoMin=-125, DeltaRho=1",
              "RhoSamples=451, RhoMin=-225, DeltaRho=1"))
rm(P,R1,R2,R3,R4,R5)
```

---

readData   *Reload Saved Datasets in different Formats.*

Description

The function offers the possibility to read a ".txt", ".dat", ".dat.gz", ".pet" or ".fif" file. Furthermore the graphics formats ".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg" and ".jpeg" are supported.

Usage

```
readData(inputfile, DebugLevel = "Normal")
```

Arguments

inputfile   (character) `inputfile` gives the name of the file to read, including the path-name to the file. The extension of the name of the file specifies the format. Currently ".txt", ".dat", ".dat.gz", ".fif", ".pet" and the graphic formats ".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg" and ".jpeg" are supported. See below to details to get more information about the formats.

DebugLevel   (character) This parameter controls the level of output. Defaults to `DebugLevel="Normal"` for a standard level output. Alternative implementations are `"Detail"` if it is desirable to show almost all output on screen or `"HardCore"` for no information at all.

Details

In the following different supported formats of `inputfile` are explained.

".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg" and ".jpeg"
For using these graphic formats the R package `adimpro` is necessary. See there, to get more information about these formats.

".txt"
Reads files in ASCII format without header, which contain only the raw data. The R-routine `read.table` is used. Returns a matrix.

".dat" and ".gz"
Read files in ASCII format that contain a header with three or four rows followed by the data. Compared to ".txt" the advantage is that is possible to save and load higher-dimensional data. Furthermore important values are saved, that are needed for Radon and the inverse Radon Transformation. If the extension ".gz" is specified, then files in ".dat" format will be read which were additionally compressed by 'gzip'.
Assuming the data are two-dimensional, then the header has to be only of the following form:

Description: char (optional)
SignalDim: int int
XYmin: double double
DeltaXY: double double
... and after the header follow the raw data

Explanation of the parameters:

Description:	Can be a short description of the data. It is possible to leave out this parameter. In this case the first line of the file begins with 'SignalDim'.
SignalDim:	In case of 2-dim data 'SignalDim' is the number of rows and columns of the array, but also a greater dimension is permissible.
XYmin:	Leftmost coordinate and lowest coordinate of the data.
DeltaXY:	Quantization steps in image, in x and y-direction.

If the dimension of data is greater than two, the number of 'SignalDim', 'XYmin' and 'DeltaXY' should be increase appropriately.

.pet

Reads a picture in ".pet" format (raw float with header) from the file e.g. `inputfile = "FileName.pet"` (binary format). The ".pet"-file has to be of the following structure: The first part contains a header and the second part the data, e.g. an image. Only the values of the parameter are saved in the header and not the parameter names.

Description:	(80-byte character) Should be a short description of image.
XSamples:	(integer) Number of rows of array.
YSamples:	(integer) Number of columns of array.
Xmin:	(float) Leftmost coordinate in original image.
Ymin:	(float) Lowest coordinate in original image.
DeltaX:	(float) Quantization steps in original image (x).
DeltaY:	(float) Quantization steps in original image (y).

The dimension of the data should be not greater than two.

The ".pet" and the sequencing ".fif" format will be used in the `iradonIT` method to store iteration-steps and to read a references image.

.fif

Reads a picture in ".fif" format (raw float with header) from the file e.g. `inputfile = "FileName.fif"` (binary format). The ".fif"-file has to be of the following structure: The first part contains a header and the second part the data, e.g. an image. At first, the values of the following parameters are saved in the file:

FIFIdType:	(integer) ID used to restore FIF:17737:'\0"\0"E"I'.
FileName:	(100-byte character) Name used for saving this image.
Description:	See above.
Date:	(10-byte character) Date (YYYY-MM-DD).
XSamples:	See above.
YSamples:	See above.
ArrayType:	(integer) Defines the format number: 1 for Real or 2 for Complex. Complex matrices are determined by $A = a_{i,j}$ where $a_{i,2n}$ is the imaginary part to the real value $a_{i,2n-1}, i = 1, \ldots, M, j = 1, \ldots, 2N, n = 1, \ldots, N$.
Xmin:	(float) See above.
Ymin:	(float) See above.
DeltaX:	(float) See above.
DeltaY:	(float) See above.
SignalMin:	(float) Lowest signal value in the array.
SignalMax:	(float) Highest signal value in the array.

After this header the data are following.

Value

In case of `inputfile="/.../FileName.txt"`, ".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg" and ".jpeg" only a matrix (image) will be returned.

In case of `inputfile="/.../FileName.dat"`, ".dat.gz", ".pet" and ".fif" a list will be returned, with

Signal   The image or the array that was read.

Header   Is a list of values specified through the file format.

In case of `inputfile="/.../FileName.dat"`, ".dat.gz" and ".pet" `$Header` contain the following list elements: Description (if ".dat" or ".dat.gz" is used then optional), SignalDim, XYmin and DeltaXY.
In case of `inputfile="/.../FileName.fif"` `$Header` contain the following list elements: FIFId-Type, FileName, Description, Date, SignalDim, ArrayType, XYmin, DeltaXY and SignalMinMax.

Author(s)

Joern Schulz, <jschulz78@web.de>.

References

Schulz, Joern, *Diploma Thesis: Analyse von PET Daten unter Einsatz adaptiver Glaettungsverfahren*, Humboldt-Universitaet zu Berlin, Institut fuer Mathematik, 2006.

See Also

[writeData](), [iradon]()

Examples

```
P <- phantom()
writeData(P, "Phantom.pet", fileOverwrite="YES" )
P.new <- readData("Phantom.pet", DebugLevel = "Normal")
viewData(P.new$Signal)
rm(P, P.new)
```

---

readSifData   *Reloaded System Matrix*

Description

The function reloaded a system matrix stored by `iradonIT`.

Usage

```
readSifData(filename, DebugLevel = "Normal")
```

Arguments

filename	(character) filename gives the name of the ".sif"-file to read, including the path-name to the file.
DebugLevel	(character) This parameter controls the level of output. Defaults to DebugLevel="Normal" for a standard level output. Alternative implementations are "Detail" if it is desirable to show almost all output on screen or "HardCore" for no information at all.

Details

If fast iterative reconstruction methods (iradonIT with parameters UseFast = 1 and KernelFileSave = 1) are used, the system matrix will be saved with the name KernelFileName. See to help of iradonIT to get detail information about these parameters. The speed of the reconstruction can obviously be increased by using of an existing system matrix.

The system matrix will be stored as a binary ".sif"-file with the following structure: At first a header is defined, then the raw data follows. The form of the header is shown below. The raw data will be stored in a ".sif"-file in the following form:

At first M integers $N_m, m = 1, \ldots, M$ are stored, which specify the number of value in each row m. Then the N_m indicies (integers) follow alternatingly for the appropriate columns and N_m accordingly (float) values for each row. Thus, a sparse matrix is defined, i.e. a (M, N)-matrix without zero-elements.

Header:

[1]	XSamples (integer)	[18]	mode (integer)
[2]	YSamples (integer)	[19]	UseFast (integer)
[3]	DeltaX (float)	[20]	ConstrainMin (float)
[4]	DeltaY (float)	[21]	ConstrainMax (float)
[5]	Xmin (float)	[22]	Alpha (float)
[6]	Ymin (float)	[23]	Beta (float)
[7]	ThetaSamples (integer)	[24]	IterationType (integer)
[8]	RhoSamples (integer)	[25]	KernelFileSave (integer)
[9]	DeltaRho (float)	[26]	SaveIterations (integer)
[10]	DeltaTheta (float)	[27]	RefFileName (200-byte character)
[11]	ThetaMin (float)	[28]	KernelFileName (200-byte character)
[12]	RhoMin (float)	[29]	InFileName (200-byte character)
[13]	LowestALevel (float)	[30]	OutFileName (200-byte character)
[14]	RadonKernel (integer)	[31]	SaveIterationsName (200-byte character)
[15]	OverSamp (integer)	[32]	M (integer)
[16]	Regularization (float)	[33]	N (integer)
[17]	Iterations (integer)		

M = RhoSamples*ThetaSamples, N = XSamples*YSamples, InFileName = "InputData" and OutFileName = "OutputData".
The other parameters are described in detail in the function iradonIT, but note that RadonKernel

will be stored as an integer ("NN" = 1, "RNN" = 2, "RL" = 3, "P1" = 4 and "P2" = 5), mode will be stored as an integer ("ART" = 0, "EM" = 1, "CG" = 2) and IterationType will be stored as an integer ("random" = 0, "cyclic" = 1).

Value

MIndexNumber   A vector of M integers $N_m, m = 1, \ldots, M$, that specify the number of values in each row m.

SparseIndex   A list of N_m indicies (integers) for the appropriate columns.

SparseSignal   A list of N_m accordingly (float) values for each row.

Header   A list of all values defined above in Header.

Author(s)

Joern Schulz, <jschulz78@web.de>.

References

Toft, Peter, *Ph.D. Thesis, The Radon Transform - Theory and Implementation*, Department of Mathematical Modelling Section for Digital Signal Processing, Technical University of Denmark, 1996.

Schulz, Joern, *Diploma Thesis: Analyse von PET Daten unter Einsatz adaptiver Glaettungsverfahren*, Humboldt-Universitaet zu Berlin, Institut fuer Mathematik, 2006.

Examples

```
## Not run:
P <- phantom(n=101)
rP <- markPoisson(P, nSample=1800000)
irP <- iradonIT(rP$rData, 101, 101, KernelFileSave=1,
            KernelFileName="SystemMatrix.sif")
rm(irP,P,rP)

# reading the sif-matrix:
SysMat <- readSifData("SystemMatrix.sif")
names(SysMat)
SysMat$Header
rm(SysMat)

## End(Not run)
```

---

rotate   *Rotates Data*

Description

The function implements the rotation of a vector, a matrix or a 3-dimensional array.

Usage

```
rotate(A, grad = 90)
```

Arguments

A   Is a vector, a matrix or a 3-dimensional array.

grad   Can be set to 0, 90, 180, 270, -90, -180 or -270. Defaults to grad = 90.

Value

Returns the rotated data.

Author(s)

Joern Schulz, <jschulz78@web.de>.

Examples

```
# Rotation of a vector
(x <- c(1:5))
rotate(x,270)
rm(x)

# Rotation of a matrix
(A <- matrix(1:15, nrow=3))
rotate(A, -90)
rm(A)

# Rotation of an array
(A <- array(1:20, c(2,5,2)))
rotate(A, 180)
rm(A)
```

---

scaleImage   *Scales an Image*

Description

Scales the minimum of an image to 0 and the mean or the maximum to 1.

Usage

```
scaleImage(image, mode = "mean")
```

Arguments

image   The raw image data.

mode   mode specifies if mean or maximum will be scaled to 1. Defaults to mode="mean".
                Alternative choice is "max".

Value

Returns the scaled data.

Author(s)

Joern Schulz, <jschulz78@web.de>.

Examples

```
(A <- matrix(-9:10,nrow=4))
scaleImage(A)
rm(A)
```

---

viewData   *Display Several Images.*

Description

This function prints 1 up to 8 images in a new window on the display or in a current window
curWindow. The function uses the R function image to display the images. Note the conditions of
this function.

Usage

```
viewData(Data, Title = NULL, curWindow = TRUE,
         colors = gray((0:255)/255), s = 1)
```

Arguments

`Data`	Has to be a matrix or a list of matrices. One, two, up to eight image are possible.
`Title`	A character or a list of titles, that specify the title of the images. Note if `Data` and `Title` are of type list then `length(Data) == length(Title)`. Defaults to `Title=NULL` (no title is uses).
`curWindow`	If `curWindow=TRUE` or `curWindow=i` then the current active window is used or the specified window i. In these cases the size of the corresponding window is used. Otherwise if `curWindow=FALSE` then a new window is open. Defaults to `curWindow=TRUE`.
`colors`	Corresponds to the parameter col of the function image. Defaults to `colors = gray((0:255)/255)`.
`s`	Scaling parameter for the size of a new window. Defaults to `s=1`.

Value

Returns the current window.

Author(s)

Joern Schulz, `<jschulz78@web.de>`.

Examples

```
P <- phantom()
R <- markPoisson(P)
viewData(list(P, R$Data, R$rData), list("Phantom",
         "Marked Poisson Data", "Simulated PET Data"))
rm(P,R)
```

---

writeData	*Write Datasets in different Formats*

Description

The function offers the possibility to write ".txt", ".dat", ".dat.gz", ".pet" or ".fif" files. Furthermore the graphic formats ".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg" and ".jpeg" are supported. The function was especially written to handle sinograms and datasets, that are processed with the functions radon, iradon and iradonIT of the R package 'PET'.

Usage

```
writeData(data, outputfile,  fileHeader = NULL, imType = "normal",
          fileOverwrite = "ASK", DebugLevel = "Normal")
```

Arguments

`data`	(matrix) The data to be written out.
`outputfile`	(character) `outputfile` naming the file to write to, including the pathname. The path has to be given relatively to the working-directory of your R-session or it contains the full path of the file. The extension of the filename specifies the format. Currently, ".txt", ".dat", ".dat.gz", ".fif", ".pet" and the graphic formats ".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg" and ".jpeg" are supported. See below to details to get more information about these formats.
`fileHeader`	(list) If the file format "is .dat", ".dat.gz", ".pet" or ".fif", then `fileHeader` is necessary. If in these cases `fileHeader=NULL`, default values will be generated dependent on `imType`. See below to details to get more information.
`imType`	(character) `imType` specifies the type of `data` and thus the default values of `fileHeader`. Default is `imType="normal"`. Additional implementation is `imType="radon"`, i.e. that the image is a sinogram (Radon transformed image).
`fileOverwrite`	(character) Control the behaviour for overwriting a file. Supported are "NO", "YES" and "ASK". If fileOverwrite = "ASK" the routine will ask you in case of an existing outputfile with this name. Default is "ASK".
`DebugLevel`	(character) This parameter controls the level of output. Defaults to `DebugLevel="Normal"` for a standard level output. Alternative implementations are `"Detail"` if it is desirable to show almost all output on screen or `"HardCore"` for no information at all.

Details

In the following different supported formats of `inputfile` are explained. For ".dat", ".dat.gz", ".pet" and ".fif" the default values dependent on `imType` are defined.

".tif", ".tiff", ".pgm", ".ppm", ".png", ".pnm", ".gif", ".jpg" and ".jpeg"
For using these graphic formats the R package `adimpro` is necessary. See there, to get more information about these formats.

".txt"
Writes files in ASCII format without header, which contain only the raw data. The R-routine `write.table` is used.

".dat" and ".gz"
Write files in ASCII format where files contain a header with three or four rows followed by the data. Compared to ".txt" the advantage is that is possible to save and load higher-dimensional data. Furthermore important values are saved, that are needed for Radon and the inverse Radon transformation. If the extension ".gz" is specified, then files in ".dat" format will be stored which are additionally compressed by 'gzip'.

Assuming the data are two-dimensional, at first `fileHeader` will be written out. Default values will be defined (dependent on `imType`) for not specified parameters. The file has only the following form:

Description: char (optional)
SignalDim: int int
XYmin: double double
DeltaXY: double double

... and after the header data will be written out

Explanation of the parameters:

Description:	Can be a short description of the data. It is possible to leave out this parameter. In this case the first line of the file begins with 'SignalDim'.
SignalDim:	In case of 2-dim data, 'SignalDim' is the number of rows and columns of the array, but also a greater dimension is permissible.
XYmin:	Leftmost coordinate and lowest coordinate of the data.
DeltaXY:	Quantization steps in image, in x and y-direction.

Defaults to:

	imType="normal"	imType="radon"
SignalDim:	dim(data)	dim(data)
XYmin:	-0.5*(dim(data)-1)	c(0,-0.5*(ncol(data)-1))
DeltaXY:	c(1,1)	c(pi/(nrow(data)),1)

If the dimension of data is greater than two, the number of 'SignalDim', 'XYmin' and 'DeltaXY' should be increased appropriately. Note in this case you must define the fileHeader.

".pet"
Writes a picture in ".pet" format (raw float with header) to the file e.g. inputfile = "FileName.pet" (binary format). The ".pet"-file has to be of the following structure: The first part contains a header and the second part the data, e.g. an image. Only the values of the parameters are saved in the header and not the parameter names.

Description:	(80-byte character) Should be a short description of image. Defaults to Description="}\eqn{\backslash 0}{\0}\code{".
XSamples:	(integer) Number of rows of array. Defaults to XSamples=nrow(data).
YSamples:	(integer) Number of columns of array. Defaults to YSamples=ncol(data).
Xmin:	(float) Leftmost coordinate in original image. If imType="normal" then defaults to Xmin= -0.5*(nrow(data)-1) or if imType="radon" then Xmin=0.
Ymin:	(float) Lowest coordinate in original image. Defaults to Ymin=-0.5*(ncol(data)-1).
DeltaX:	(float) Quantization steps in original image (x). If imType="normal" then defaults to DeltaX=1 or if imType="radon" then DeltaX=pi/(nrow(data)).
DeltaY:	(float) Quantization steps in original image (y).

Defaults to `DeltaY=1`.

The dimension of the data should be not greater than two.
The ".pet" and the sequencing ".fif" format will be used in the `iradonIT` method to store iteration-steps and to read a references image.

".fif"

Writes a picture in ".fif" format (raw float with header) to the file e.g. `inputfile = "FileName.fif"` (binary format). The ".fif"-file has to be of the following structure: The first part contains a header and the second part the data, e.g. an image. At first the values of the following parameters are saved in the file:

FIFIdType:	(integer) ID used to restore FIF:17737:'\0"\0"E"I'. Defaults to `FIFIdType=17737`.
FileName:	(100-byte character) Name used for saving this image. Defaults to `FileName=outputfile`.
Description:	See above.
Date:	(10-byte character) Date (YYYY-MM-DD). Defaults to `Date=as.character(Sys.Date())`.
XSamples:	See above.
YSamples:	See above.
ArrayType:	(integer) Defines the format number: 1 for Real or 2 for Complex. Complex matrices are determined by $A = a_{i,j}$ where $a_{i,2n}$ is the imaginary part to the real value $a_{i,2n-1}$, $i = 1, \ldots, M$, $j = 1, \ldots, 2N$, $n = 1, \ldots, N$. Defaults to `ArrayType=1`.
Xmin:	(float) See above.
Ymin:	(float) See above.
DeltaX:	(float) See above.
DeltaY:	(float) See above.
SignalMin:	(float) Lowest signalvalue in array. Defaults to `SignalMax=max(data)`.
SignalMax:	(float) Highest signalvalue in array. Defaults to `SignalMax=max(data)`.

After this header `data` are following.

Value

`writeData` creates a file, but returns nothing.

Author(s)

Joern Schulz, <jschulz78@web.de>.

References

Schulz, Joern, *Diploma Thesis: Analyse von PET Daten unter Einsatz adaptiver Glaettungsverfahren*, Humboldt-Universitaet zu Berlin, Institut fuer Mathematik, 2006.

See Also

[readData](#), [radon](#)

Examples

```
## Not run:
P <- phantom(addIm="blurred1")
rP <- radon(P, RhoSamples=401)
irP <- iradon(rP$rData, 257, 257)
# Saving 'P' as a 'jpeg'-file
writeData(P, "Phantom.jpeg")
# Saving 'rP' as a 'pet'-file
writeData(rP$rData, "RadonPhantom.pet", fileHeader=rP$Header)
# Saving 'irP' as a 'dat.gz'-file'
writeData(irP$irData, "RecPhantom.dat.gz", fileHeader=irP$Header)
rm(irP,P,rP)

Image1 <- readData("Phantom.jpeg")
Image2 <- readData("RadonPhantom.pet")
Image3 <- readData("RecPhantom.dat.gz")
viewData(list(Image1,Image2$Signal,Image3$Signal))
rm(Image1,Image2,Image3)

## End(Not run)
```

Index