

Package ‘adimpro’

October 12, 2009

Version 0.7.3

Date 2009-10-12

Title Adaptive Smoothing of Digital Images

Author Karsten Tabelow <tabelow@wias-berlin.de>, Joerg Polzehl <polzehl@wias-berlin.de>

Maintainer Karsten Tabelow <tabelow@wias-berlin.de>

Depends R (>= 2.6.0), grDevices

Requires Image Magick (for reading non PPM format), ddraw (for reading RAW images).

Description This package implements tools for manipulationg digital images and the Propagation Separation approach by Polzehl and Spokoiny (2006) for smoothing digital images.

License GPL (>= 2)

Copyright This package is Copyright (C) 2006-2009 Weierstrass Institute for Applied Analysis and Stochastics.

URL <http://www.wias-berlin.de/software/imaging/>

Repository CRAN

Date/Publication 2009-10-12 10:42:13

R topics documented:

adimpro	2
adimpro.options	4
adjust.image	5
awsimage	6
awsraw	10
clip.image	11
colorspace	12
combine	14
develop.raw	15

edges	17
extract.image	18
extract.info	19
extract.ni	20
imganiso2D	20
mask.create	21
plot.adimpro	22
rotate.image	24
segment	25
show.image	28
shrink.image	29
summary.adimpro	30
write.image	31
write.raw	32

Index **34**

adimpro *I/O Functions*

Description

Create image objects of class "adimpro" from arrays, RAW-format files and other image formats.

Usage

```
read.raw(filename, type="PPM",
         wb="CAMERA", cspace="Adobe", interp="Bilinear", maxrange=TRUE,
         rm.ppm=TRUE, compress=TRUE)
read.image(filename, compress=TRUE)
make.image(x, compress=TRUE, gammatype="None", whitep = "D65", cspace="Adobe", scale
```

Arguments

filename	file name
x	Array or matrix containing RGB or greyscale values in the range (0,1) or (0,65535).
type	option settings for ddraw. default "PPM". type="png" allows to read grey-value png images as RAW-data (used as internal solution to store RAW information)
wb	white balance. default "CAMERA"
cspace	defines the output color space, default "sRGB" (sRGB D65), alternatives are "RAW" (Camera specific), "Adobe" (Adobe 1998 D65), "wGamut" (Wide Gamut D65), "kodak" (Kodak ProPhoto D65) and "XYZ", see manpages of ddraw.
interp	defines the interpolation method, default "Bilinear", Alternatives are "VNG", "AHD", "FourC" (Four color interpolation) and "Halfsize", see manpages of ddraw. "VNG" seems to provide the smallest spatial correlations.

<code>maxrange</code>	If TRUE increase range of values to maximum.
<code>rm.ppm</code>	remove intermediate tmp file? default TRUE
<code>gammatype</code>	character, determines the type of gamma correction within the image. "ITU" stands for ITU-R BT.709-3 as e.g. used by <code>dcrw</code> . Alternatives recognized within the package are "None", "sRGB" and "CIE" (CIE L*). Please specify if you know that your image is not gamma corrected using ITU-R BT.709-3.
<code>whitept</code>	White point in xyY space. Can be given as one of (character) <code>c("A", "B", "C", "E", "D50", "D55", "D65")</code> or as a two element numeric vector of chromatic xy coordinates. "D65" corresponds to the default white point of "sRGB" and "Adobe" RGB-spaces.
<code>compress</code>	logical, determines if image data are stored in raw-format.
<code>scale</code>	"Original" scales to $(0, \max(\text{img}\$img))$ if $\min(\text{img}\$img) < 0$, otherwise keeps the original scale. "Maxcontrast" scales each channel to maximum contrast
<code>xmode</code>	<code>xmode</code> determines how to interpret the values in <code>x</code> if $\text{length}(\text{dim}(x)) == 3$. Implemented are <code>xmode="RGB"</code> (default) and <code>xmode="HSI"</code>

Details

If ImageMagick is available on the system, `read.image` reads any of the following image file formats: `c("tif", "tiff", "pgm", "ppm", "png", "pnm", "gif", "jpg", "jpeg")` converts it into a temporary "pgm" or "ppm" file. This file is removed after reading the image. If ImageMagick is not available only "pgm", "ppm" and "pnm" formats can be processed.

If `dcrw` is available on the system, `read.raw` reads many RAW formats. `type` sets options to `dcrw`: "PPM" sets "-4", "RAW" sets "-4 -d", "HALFSIZE" sets "-h", "INFO" sets "-i -v". `wb` indicates, which white balance should be used: "NONE", "AUTO", "CAMERA".

Functions `read.raw(file, type="RAW")` and `read.image(file)` provide identical results on png-mages. If the result is a color, greyvalued or RAW image depends on the content of the comment associated with the png-image.

`make.image` converts an appropriate 2 or 3 dimensional array to an image object of class "adimpro".

Value

object of class "adimpro" containing the image. The object has the following components:

<code>img</code>	array containing the color values in the color space specified by <code>value\$type</code> .
<code>type</code>	the color space.
<code>depth</code>	color depth, here "16bit".
<code>dim</code>	vector of length 2 containing the number of pixel in horizontal and vertical direction.
<code>file</code>	the argument <code>file</code> identifying the image.
<code>cspace</code>	the type of rgb space used, as specified by <code>cspace</code> .
<code>interp</code>	interpolation applied by <code>dcrw</code> , as specified by <code>interp</code> .
<code>gamma</code>	has a gamma correction been applied, here FALSE for <code>read.raw</code> and TRUE for <code>read.image</code>

`gammatype` type of gamma correction `read.image`.
`wb` type of white balance, as specified by `wb`.
`compressed` image data are stored as raw-vector (TRUE) or array of integers (FALSE).

Note

The function `read.raw` requires `dcrw` to be installed.

Author(s)

Karsten Tabelow [⟨tabelow@wias-berlin.de⟩](mailto:tabelow@wias-berlin.de) and Joerg Polzehl [⟨polzehl@wias-berlin.de⟩](mailto:polzehl@wias-berlin.de)

See Also

[read.image](#)

Examples

```
## Not run: demo(io)
```

`adimpro.options` *Set parameters for graphical display*

Description

On systems capable of X11 the function sets the X11-type (preferably "Xlib"). It also sets a default size for graphical displays opened by functions from the package.

Usage

```
adimpro.options(xsize = NULL, ysize = NULL)
```

Arguments

`xsize` display width in pt
`ysize` display height in pt

Details

The function assigns the specified values to as a list to the variable name ".adimpro". This variable is, if it exists, evaluated by several other functions.

On some systems the default `X11.options()$type` "cairo" leads to significant slower image display. You may try to use `X11.options(type="Xlib")` instead. To automatically choose this option set the system environment variable `R_X11type` (`setenv R_X11type Xlib` or `export R_X11type=Xlib`) before loading the package.

`alg` determines the approximation for the gamma correction. Select 1 for fastest computation and 3 for maximum accuracy, or 2 for a compromise.

`compress` logical, determines if image data are stored in raw-format.

Details

This function adjusts color channels and applies gamma correction (if applicable).

If `color.par$red` or `color.par$blue` or `color.par$brightness` differ from 1.0 the corresponding channels are multiplied with the provided values. Saturated values are set to 1.

If `img$gamma==FALSE`, perform gamma correction with `color.par$ga` and `color.par$bp`. `alg` chooses between three different computing algorithms (approximations) with increasing computation time and precision (`alg` is 1,2, or 3).

Value

Adjusted image object of class "adimpro".

Author(s)

Karsten Tabelow [⟨tabelow@wias-berlin.de⟩](mailto:tabelow@wias-berlin.de) and Joerg Polzehl [⟨polzehl@wias-berlin.de⟩](mailto:polzehl@wias-berlin.de)

See Also

[show.image](#), [write.image](#)

Examples

```
## Not run: demo(color)
```

awsimage

Propagation-Separation approach for smoothing of 2D images

Description

This functions implement the Propagation-Separation approach (local constant and local polynomial model) for smoothing images. Function `awsaniso` uses anisotropic location weights. This is done by evaluating local gradient estimates obtained from the actual estimated color values.

Usage

```
awsimage(object, hmax=4, aws=TRUE, varmodel=NULL, ladjust=1.25,
          mask=NULL, xind = NULL, yind = NULL,
          wgths=c(1,1,1,1), scorr=TRUE,
          lkern="Plateau", plateau=NULL, homogen=TRUE, earlystop=TRUE,
          demo=FALSE, graph=FALSE,
          max.pixel=4.e2, clip = FALSE, compress=TRUE)
awspimage(object, hmax=12, aws=TRUE, degree=1, varmodel = NULL,
```

```

ladjust=1.0, xind = NULL, yind = NULL,
wghts=c(1,1,1,1), scorr= TRUE,
lkern="Plateau", plateau=NULL, homogen=TRUE, earlystop=TRUE,
demo=FALSE, graph=FALSE,
max.pixel= 4.e2, clip = FALSE, compress=TRUE)
awsaniso(object, hmax = 4, g = 3, rho = 0, aws = TRUE, varmodel = NULL,
ladjust = 1, xind = NULL, yind = NULL, wghts = c(1, 1, 1, 1),
scorr = TRUE, lkern = "Triangle", demo = FALSE, graph = FALSE,
satexp = 0.25, max.pixel = 400, clip = FALSE, compress = TRUE)

```

Arguments

<code>object</code>	Image object, class "adimpro", as from <code>read.image</code> , <code>read.raw</code> , or <code>make.image</code> .
<code>hmax</code>	Maximum bandwidth to use in the iteration procedure.
<code>g</code>	Bandwidth for anisotropic smoothing gradient estimates, preferably $g \geq 3$ for images with line type texture and small $g \approx 1$ for improving edges between homogeneous regions (function <code>awsaniso</code> only).
<code>rho</code>	Regularization parameter for anisotropic smoothing gradient estimates, preferably $\rho = 0$ for images with line type texture and large $\rho \approx 3$ for improving edges between homogeneous regions. (function <code>awsaniso</code> only)
<code>aws</code>	(logical). If TRUE the propagation - separation (PS) approach from Polzehl and Spokoiny (2006) is used. <code>aws=FALSE</code> turns off the statistical penalty resulting in a nonadaptive kernel estimate using a kernel with bandwidth <code>hmax</code> .
<code>degree</code>	Degree of the local polynomial model for <code>awspimage</code> . 0, 1, or 2 only.
<code>varmodel</code>	<code>varmodel</code> specifies how variances are to be estimated. This can be a homogeneous variance estimate (<code>varmodel="None"</code>) assuming uncorrelated errors (both spatial and between channels). Alternatives are an adaptive homogeneous or linear (function of the mean) variance estimate that depends on estimated correlations and on residuals from the last iteration step. The default <code>varmodel=NULL</code> corresponds to <code>varmodel == "Linear"</code> if <code>img\$gamma==FALSE</code> and <code>varmodel == "Constant"</code> otherwise.
<code>ladjust</code>	adjustment factor for lambda (≥ 1). Default values for lambda are selected for Gaussian distributions and default settings of parameters <code>lkern</code> and <code>plateau</code> . Skewed or heavy tailed distributions may require slightly larger values for lambda to meet the propagation condition. <code>ladjust</code> allows to increase lambda in such situations.
<code>mask</code>	logical array of the same size as the image or NULL (default). Smoothing is restricted to the smallest rectangle including all pixel where <code>mask==TRUE</code> and restricts computations to these pixel. This need not be a connected area (Typical usage: smooth all bright regions)! Only used if <code>is.null(xind) && is.null(yind)</code> . Inactive if <code>mask==NULL</code> . Can only be used if <code>varmodel="None"</code> .
<code>xind, yind</code>	Restrict smoothing to rectangular area defined by pixel indices <code>xind, yind</code> in x- and y-direction. Full range if NULL (default).
<code>wghts</code>	allows to weight the information from different (up to 4) color channels. The weights are used in the statistical penalty of the PS-procedure. Note that lambda-values are selected for <code>wghts==c(1, 1, 1, 1)</code> , please use parameter <code>ladjust</code> to set an appropriate value.

<code>scorr</code>	(logical). Specifies whether spatial correlation is to be estimated. Defaults to TRUE. Is set to FALSE if <code>mask</code> is not NULL.
<code>lkern</code>	Specifies the location kernel. Defaults to "Triangle", other choices are "Quadratic", "Cubic" and "Uniform". The use of "Triangle" corresponds to the Epanechnikov kernel nonparametric kernel regression.
<code>plateau</code>	Extension of the plateau in the statistical kernel. Can take values from (0,1), defaults to 0.25.
<code>homogen</code>	If TRUE the algorithm determines, in each design point i , a circle of maximum radius, such that the statistical penalty $s_{\{ij\}}$ for all points j within the circle is less than the value specified in <code>plateau</code> . In subsequent iteration steps the statistical penalty for such points is set to zero. This is only used if <code>plateau</code> >0. This results in more stable intermediate estimates and in a smoother reconstruction. <code>homogen</code> =TRUE leads to increased memory requirements.
<code>earlystop</code>	If TRUE the algorithm determines, in each design point i , a circle of minimal radius, such that the circle includes all point j with positive weights $w_{\{ij\}}$. If this radius is considerably smaller than the actual bandwidth then the estimate in point i is fixed. This should considerably reduce computing time in case of large <code>hmax</code> . <code>earlystop</code> =TRUE slightly increases memory requirements.
<code>demo</code>	(logical). If <code>demo</code> =TRUE the function pauses after each iteration. Defaults to FALSE.
<code>graph</code>	(logical). If <code>graph</code> =TRUE intermediate results are illustrated after each iteration step. Defaults to FALSE.
<code>max.pixel</code>	Maximum dimension of images for display if <code>graph</code> =TRUE. If the true dimension is larger, the images are downscaled for display. See also show.image .
<code>satexp</code>	exponent used for scaling saturation in anisotropy visualization (function <code>awsaniso</code> only)
<code>clip</code>	(logical). If TRUE a clipping region is selected, see clip.image , using the information contained in <code>xind</code> or <code>yind</code> . If both are NULL a clipping region can be defined by left mouse clicks. The image object is reduced to the clipping region before smoothing.
<code>compress</code>	logical, determines if image data are stored in raw-format.

Details

The function implements the Propagation-Separation (PS) approach to nonparametric smoothing (formerly introduced as Adaptive Weights Smoothing) for varying coefficient likelihood (`awsimage`) and local polynomial (`awspimage`) models for greyscale and color images.

The distribution of grey (color) values is considered to be Gaussian. Noise can be colored.

The numerical complexity of the procedure is mainly determined by `hmax`. The number of iterations is $2 \cdot \log(hmax) / \log(1.25)$. Complexity in each iteration step is $Const \cdot hakt \cdot n$ with `hakt` being the actual bandwidth in the iteration step and `n` the number of pixels. `hmax` determines the maximal possible variance reduction.

All other parameters of the approach only depend on the specified values for `skern`/`lkern` and are therefore set internally to meaningful default values.

For a detailed description of the procedure see references below.

The script used to control the values of parameter `lambda` is stored in directory `inst/adjust`.

Value

Object of class "adimpro"

<code>img</code>	Contains the reconstructed image.
<code>ni</code>	Contains the sum of weights, i.e. $\text{trace}(W_i)$, in all grid points i .
<code>ni0</code>	Contains the maximum sum of weights for a nonadaptive kernel estimate with the same bandwidth.
<code>hmax</code>	Bandwidth used in the last iteration.
<code>call</code>	The arguments of the function call.
<code>varcoef</code>	Estimated coefficients in the variance model for the color channels, if <code>varmodel</code> is "Constant" or "Linear".
<code>wghts</code>	The weights used for the color channels.
<code>scorr</code>	Estimated spatial correlations for each channel, if <code>scorr=TRUE</code>
<code>chcorr</code>	Estimated correlations between color channels, if <code>scorr=TRUE</code>

Author(s)

Karsten Tabelow [⟨tabelow@wias-berlin.de⟩](mailto:tabelow@wias-berlin.de) and Joerg Polzehl [⟨polzehl@wias-berlin.de⟩](mailto:polzehl@wias-berlin.de)

References

Polzehl and Spokoiny (2006). Propagation-Separation Approach for Local Likelihood Estimation. *Probability Theory and Related Fields*. 3 (135) 335 - 362.

Polzehl and Spokoiny (2005). Structural adaptive smoothing adaptive smoothing by Propagation-Separation-methods. WIAS-Preprint No. 1068.

Polzehl, J. and Tabelow, K. (2007). Adaptive smoothing of digital images, *Journal of Statistical Software* 19 (1).

See Also

[read.image](#), [read.raw](#), [make.image](#), [show.image](#), [clip.image](#)

Examples

```
## Not run: demo(awsimage)
```

awsraw

*Smoothing and demosaicing of RAW images***Description**

The function integrates smoothing and demosaicing of RAW image data.

Usage

```
awsraw(object, hmax = 4, aws = TRUE, wb = c(1, 1, 1), cspace = "Adobe", ladjust = 1,
lkern = "Triangle", graph = FALSE, max.pixel = 400, compress = TRUE)
```

Arguments

object	an object of class <code>adimpro</code> containing the RAW image data. See read.raw for creating such objects.
hmax	maximal bandwidth to use in the smoothing algorithm.
aws	use adaptive weights if <code>aws==TRUE</code> .
wb	Vector containing factors for the three color channels, allows to change the white balance.
cspace	Color space of the result,
ladjust	Factor for the critical value λ . Defaults to 1, smaller values increase sensitivity but may result in isolated noisy pixel. Values larger than 1 give smoother up to cartoon like results.
maxrange	If <code>TRUE</code> increase range of values to maximum.
lkern	Specifies the location kernel. Defaults to "Triangle", other choices are "Quadratic", "Cubic" and "Uniform". The use of "Triangle" corresponds to the Epanechnikov kernel nonparametric kernel regression.
graph	(logical). If <code>graph=TRUE</code> intermediate results are illustrated after each iteration step. Defaults to <code>FALSE</code> .
max.pixel	Maximum dimension of images for display if <code>graph=TRUE</code> . If the true dimension is larger, the images are downscaled for display. See also show.image .
compress	logical, determines if image data are stored in raw-format.

Details

Adaptive smoothing is performed on the original RAW data, restricting positive weights to pixel corresponding to the same color channel. Noise is assumed to have a variance depending linearly on the mean. Weights are determined by weighted distances between color vectors. These color vectors are obtained by demosaicing that is applied to the smoothed RAW data after each iteration of the smoothing algorithm. The demosaicing algorithm is a 3D generalized median, see `method="Median4"` in function [develop.raw](#).

Value

Object of class "adimpro"

img	Contains the reconstructed image.
ni	Contains the sum of weights, i.e. $\text{trace}(\bar{W}_i)$, in all grid points i .
ni0	Contains the maximum sum of weights for a nonadaptive kernel estimate with the same bandwidth.
hmax	Bandwidth used in the last iteration.
call	The arguments of the function call.
varcoef	Estimated coefficients in the linear variance model for the color channels.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

References

Polzehl, J. and Tabelow, K. (2007). Adaptive smoothing of digital images, Journal of Statistical Software 19 (1).

See Also

[read.raw](#), [awsimage](#), [make.image](#), [show.image](#), [clip.image](#)

Examples

```
## Not run: demo(raw)
```

clip.image	<i>Create an image by clipping</i>
------------	------------------------------------

Description

The function allows to define a clipping region by arguments `xind` and `yind` or interactively by mouseclicks. A new image is created by cutting out the clipping region.

Usage

```
clip.image(img, xind = NULL, yind = NULL, compress=NULL, ...)
```

Arguments

img	Object of class "adimpro" as created by read.image , read.raw or make.image .
xind	<code>xind</code> defines the horizontal extension of the clipping region.
yind	<code>yind</code> defines the vertical extension of the clipping region.
compress	image data are stored as raw-vector (TRUE) or array of integers (FALSE). <code>compress=NULL</code> keeps the format used in <code>img</code> .
...	additional arguments to show.image can be passed here.

Details

If both `xind==NULL` and `yind==NULL`, `show.image` is called and the clipping region can be set by two left mouse clicks to opposite corners.

Value

An object of class "adimpro".

Author(s)

Karsten Tabelow (tabelow@wias-berlin.de) and Joerg Polzehl (polzehl@wias-berlin.de)

See Also

`read.image`, `read.raw`, `make.image`, `awsimage`, `awspimage`

Examples

```
## Not run: demo(manipulate)
```

colorspace

Color Space Conversion

Description

Color space conversion functions for RGB, YUV, YIQ, XYZ, and HSI.

Usage

```
rgb2grey(obj, compress=TRUE)
rgb2hsi(obj)
hsi2rgb(obj, cspace = "Adobe", compress=TRUE)
rgb2yuv(obj)
yuv2rgb(obj, cspace = "Adobe", compress=TRUE)
rgb2yiq(obj)
yiq2rgb(obj, cspace = "Adobe", compress=TRUE)
rgb2xyz(obj)
xyz2rgb(obj, cspace = "Adobe", black= 0, exposure=1, compress=TRUE)
```

Arguments

<code>obj</code>	an object of class "adimpro", <code>obj\$type</code> should coincide with the first three letters of the function name, that is <code>obj\$img</code> contains color values in the corresponding color space.
<code>cspace</code>	Target color space, one of <code>c("sRGB", "Adobe", "wGamut", "kodak", "xyz")</code> . For function <code>xyz2rgb</code> also <code>c("yuv", "yiq")</code> are implemented.

<code>exposure</code>	Multiplicative factor for all color channels (in xyz or rgb spaces). Applied in linear color space, i.e. if the image is gamma corrected the gamma correction is reversed first.
<code>black</code>	Adjustment for black color. Color values with luminance \leq black will be assigned to black in RGB.
<code>compress</code>	logical, determines if image data are returned in raw-format.

Details

The functions convert an image `obj` from one color space into another.

`rgb2grey` converts from RGB to GREYSCALE.

`rgb2hsi` and `hsi2rgb` convert from RGB to HSI and vice versa.

`rgb2yuv` and `yuv2rgb` convert from RGB to YUV and vice versa.

`rgb2yiq` and `yiq2rgb` convert from RGB to YIQ and vice versa.

`rgb2xyz` and `xyz2rgb` convert from RGB to CIE XYZ and vice versa.

Conversion to XYZ, YIQ, YUV and HSI involves an inverse gamma correction if required.

Value

an object of class "adimpro", with `value$type` specifying the color space (last three letters of the function name or 'greyscale' for `rgb2grey`) and `value$img` containing the color values.

Author(s)

Karsten Tabelow [⟨tabelow@wias-berlin.de⟩](mailto:tabelow@wias-berlin.de) and Joerg Polzehl [⟨polzehl@wias-berlin.de⟩](mailto:polzehl@wias-berlin.de)

References

Gonzalez, R.C., and Woods, R.E. (2002). Digital Image Processing. Prentice Hall.

Polzehl, J., and Tabelow, K. (2007). Adaptive smoothing of digital images, Journal of Statistical Software 19 (1).

Examples

```
## Not run: demo(color)
```

 combine

Pixelwise operations on a pair of images

Description

The function allows to perform pixelwise operations, specified by a supplied function, on a pair of images.

Usage

```
combine(img1, img2, fun = "+", rescale = TRUE, compress = TRUE, gammatype = "None",
```

Arguments

<code>img1</code>	image, object of class <code>adimpro</code>
<code>img2</code>	image, object of class <code>adimpro</code> , need to have the same dimension as <code>img1</code>
<code>fun</code>	A function or primitive of two (or more) arguments specifying the operation. The first argument corresponds to grey/color-values in <code>img1</code> , the second to <code>img2</code> . Auxiliary parameters can be passed through "..."
<code>rescale</code>	logical: if <code>TRUE</code> the resulting image is rescaled to fit into the range of possible grey/color-values, if <code>FALSE</code> values outside the range are truncated.
<code>compress</code>	logical, determines if image data are stored in raw-format.
<code>gammatype</code>	character, determines the type of gamma correction within the image. "ITU" stands for ITU-R BT.709-3 as e.g. used by <code>dcrw</code> . Alternatives recognized within the package are "None", "sRGB" and "CIE" (CIE L*). Please specify if you know that your image is not gamma corrected using ITU-R BT.709-3.
<code>whitep</code>	White point in <code>xyY</code> space. Can be given as one of (character) <code>c("A", "B", "C", "E", "D50", "D55", ...)</code> or as a two element numeric vector of chromatic <code>xy</code> coordinates. "D65" corresponds to the default white point of "sRGB" and "Adobe" RGB-spaces.
<code>cspace</code>	defines the output color space, default "sRGB" (sRGB D65), alternatives are "RAW" (Camera specific), "Adobe" (Adobe 1998 D65), "wGamut" (Wide Gamut D65), "kodak" (Kodak ProPhoto D65) and "XYZ", see manpages of <code>dcrw</code> .
<code>xmode</code>	<code>xmode</code> determines how to interpret the values in <code>x</code> if <code>length(dim(x)) == 3</code> . Implemented are <code>xmode="RGB"</code> (default) and <code>xmode="HSI"</code>
...	additional parameters for function <code>fun</code>

Details

There are two major applications for this function. First it allows to add noise to an image by first creating an image that contains the noise and then adding this image using `fun="+"`. Second it offers a way to replace parts of an image, see examples.

Value

object of class "adimpro" containing the image. The object has the following components:

img	array containing the color values in the color space specified by <code>value\$type</code> .
type	the color space.
depth	color depth, here "16bit".
dim	vector of length 2 containing the number of pixel in horizontal and vertical direction.
file	the argument <code>file</code> identifying the image.
cspace	the type of rgb space used, as specified by <code>cspace</code> .
interp	interpolation applied by <code>dcrw</code> , as specified by <code>interp</code> .
gamma	has a gamma correction been applied, here FALSE for <code>read.raw</code> and TRUE for <code>read.image</code>
gammatype	type of gamma correction <code>read.image</code> .
wb	type of white balance, as specified by <code>wb</code> .
compressed	image data are stored as raw-vector (TRUE) or array of integers (FALSE).

Author(s)

Karsten Tabelow [⟨tabelow@wias-berlin.de⟩](mailto:tabelow@wias-berlin.de) and Joerg Polzehl [⟨polzehl@wias-berlin.de⟩](mailto:polzehl@wias-berlin.de)

See Also

[make.image](#)

Examples

```
## Not run: demo(combine)
```

develop.raw

Create a color image from RAW image data.

Description

The function generates a color image from RAW image data.

Usage

```
develop.raw(object, method = "BILINEAR", wb = c(1, 1, 1), maxrange= TRUE, compress
```

Arguments

object	An object of class "adimpro" containing RAW image data, e.g. created by function <code>read.raw</code> . Such an object is characterized by <code>object\$type=="RAW"</code> .
method	Method to fill missing color values. Currently implemented are <code>method="HALF"</code> (reduce image size by factor of 2), <code>method="FULL"</code> (use color from neighbor within (2x2) Bayer mask), <code>method="BILINEAR"</code> (bilinear interpolation), <code>method="Median4"</code> and <code>method="Median16"</code> , the latter two being based on L1-MM over shifted Bayer masks. <code>method="Median16"</code> delivers much smoother results, but is considerably slower than the other methods.
wb	numerical vector of length 3 containing multiplicative factors for the three color channels.
maxrange	If TRUE increase range of values to maximum.
compress	logical, determines if image data are stored as <code>raw</code> .

Details

RAW image data usually contain only one color value at each pixel, with colors arranged in a so called Bayer mask. Converting RAW images into color images requires to fill the missing entries in the color channels.

Value

object of class "adimpro" containing the image.

Note

The function requires `dcraw` to be installed.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

[read.image](#)

Examples

```
## Not run: demo(io)
## Not run: demo(raw)
```

Description

Edge detection using Laplacian, Sobel, or Robert Cross filter.

Usage

```
edges(img, type = "Laplacian", ltype=1, abs=FALSE)
```

Arguments

img	an object of class "adimpro".
type	type of edges detection filter. "Laplacian" (default), "Sobel" , or "Robertcross".
ltype	type of laplacian filter. 1,2,3, or 4
abs	take absolute values of results. This has only an effect for tyoe="Laplacian"

Details

This function applies the Laplacian, Sobel, or Robert Cross filter to the input image `img`. The filter is applied to each color channel separately. `ltype` determines the different matrices for Laplacian filter used in the literature. `ltype == 1` will use:

```
conv <- matrix(c(-1,-1,-1,-1,-1, -1,-1,-1,-1,-1, -1,-1,24,-1,-1, -1,-1,-1,-1,-1, -1,-1,-1,-1,-1), 5, 5)
```

`ltype == 2` will use:

```
conv <- matrix(c(0,-1,0,-1,4,-1,0,-1,0), 3, 3)
```

`ltype == 3` will use:

```
conv <- matrix(c(-1,-1,-1,-1,8,-1,-1,-1,-1), 3, 3)
```

`ltype == 4` (default) will use:

```
conv <- matrix(c(1,-2,1,-2,4,-2,1,-2,1), 3, 3)
```

Value

Array containing the values for the edge detector in each pixel and color channel.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

References

Gonzalez, R.C., and Woods, R.E. (2002). Digital Image Processing. Prentice Hall.

Examples

```
## Not run: demo(manipulate)
## Not run: demo(awspimage)
```

extract.image	<i>Extract image data from adimpro object</i>
---------------	---

Description

Extract image data from adimpro object

Usage

```
extract.image(object)
```

Arguments

object adimpro object

Value

array or matrix of image data

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

[make.image](#)

Examples

```
## Not run: demo(io)
## Not run: demo(manipulate)
```

extract.info	<i>Extract EXIF information and additional characteristics from an object of class "adimpro".</i>
--------------	---

Description

Extract EXIF information and additional characteristics from an object of class "adimpro".

Usage

```
extract.info(object, what = "Bayer")
```

Arguments

object	an object of class "adimpro" or a character string that was written as a comment by functions write.image or write.raw .
what	A character string specifying which information is to be extracted. Currently implemented are "Bayer" (Bayer mask), "Daymulti" (Daylight multipliers), "Cammulti" (camera multipliers), "Camera" (camera model), "Isize" (Image size), "Osize" (image output size), "File" (source filename), "Interpolation" (Interpolation used to convert RAW to RGB), "Gammatype" (type of gamma correction), "WhiteBalance" (white balance), "WhitePoint" (white point), "Type" (image type), "xind" (horizontal subindex used by function clip.image) and "yind" (vertical subindex used by function clip.image).

Details

This function is mainly used to access and manipulate information when reading or developing images. A more comprehensive overview is provided by function [summary](#).

Value

A character string or integer vector depending on the argument "what".

Note

ImageMagick has to be installed on the system to write "tif", "tiff", "png", "gif", "jpg" or "jpeg" files.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

[read.raw](#), [read.image](#), [develop.raw](#), [summary](#)

`extract.ni`*extract information about effective size of neighborhoods*

Description

The function allows to extract information about the effective size of neighborhoods used in each pixel from objects generated by `awsimage` or `awspimage`. The result is converted into a greyscale image.

Usage

```
extract.ni(object, gamma = FALSE, compress = TRUE)
```

Arguments

<code>object</code>	object returned from <code>awsimage</code> or <code>awspimage</code> .
<code>gamma</code>	logical, value of component <code>gamma</code> in the returned image.
<code>compress</code>	logical, if <code>TRUE</code> the returned image will be compressed.

Value

an object of class "adimpro".

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

`awsimage`, `awspimage`, `show.image`, `write.image`

`imganiso2D`*create an image that visualizes anisotropy*

Description

The function creates an object of class `adimpro` that visualizes anisotropy information using the HSI color space for main direction of anisotropy (H), maximum eigenvalue^{satexp} (S) and log(excentricity) (I).

Usage

```
imganiso2D(x, satexp = 0.25, g=3, rho=0)
```

Arguments

x	a field of 2D tensors (Dimension $c(3, n1, n2)$) or an object of class "adimpro".
satexp	exponent for maximum eigenvalue in saturation channel. Determines the contrast in this channel.
g	Bandwidth for anisotropic smoothing gradient estimates, preferably $g \geq 3$ for images with line type texture and small $g \approx 1$ for improving edges between homogeneous regions.
rho	Regularization parameter for anisotropic smoothing gradient estimates, preferably $\rho = 0$ for images with line type texture and large $\rho \approx 3 * variance$ for improving edges between homogeneous regions.

Value

an object (image) of class adimpro.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

References

Polzehl, J. and Tabelow, K. (2007). Adaptive smoothing of digital images, Journal of Statistical Software 19 (1).

See Also

[awsaniso](#), [read.image](#), [read.raw](#), [make.image](#), [show.image](#), [clip.image](#)

Examples

```
## Not run: demo(awsimage)
## Not run: demo(manipulate)
```

mask.create

Create a mask for use within function awsimage

Description

Select part of image according to greyscale (or color) value.

Usage

```
mask.create(img,
            range1 = c(0, 1), range2 = c(0, 1), range3 = c(0, 1),
            locate = TRUE)
```

Arguments

img	Image object of class "adimpro", usually the result returned from <code>read.image</code> , <code>read.raw</code> , or <code>make.image</code> .
range1	Range of gray values or values in the first color channel. <code>mask</code> is set to <code>FALSE</code> for all pixel with values outside this range. Defaults to <code>c(0, 1)</code> .
range2	Range of values in the second color channel. <code>mask</code> is set to <code>FALSE</code> for all pixel with values outside this range. Defaults to <code>c(0, 1)</code> .
range3	Range of values in the third color channel. <code>mask</code> is set to <code>FALSE</code> for all pixel with values outside this range. Defaults to <code>c(0, 1)</code> .
locate	(logical). If <code>TRUE</code> (default), the image is displayed and two opposite corners of a rectangular region can be selected using the mouse (call of <code>locator(2)</code>).

Details

Function to create a mask. The returned object `mask` can be used to restrict computations in function `awsimage` to a region characterized by `mask==TRUE`.

Value

logical matrix of image dimension.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

References

Polzehl, J., and Tabelow, K. (2007). Adaptive smoothing of digital images, *Journal of Statistical Software* 19 (1).

See Also

[awsimage](#)

plot.adimpro

I/O functions

Description

Visualize image data.

Usage

```
## S3 method for class 'adimpro':
plot(x, new=FALSE, gammatype = NULL, cspace = NULL,
      whitep = NULL, temp = NULL, black = 0, exposure = 1,...)
```

Arguments

<code>x</code>	image object of class "adimpro"
<code>new</code>	should new X11() be opened? default FALSE
<code>gammatype</code>	character, determines the type of gamma correction within the image. "ITU" stands for ITU-R BT.709-3 as e.g. used by dcrw. Alternatives recognized within the package are "None", "sRGB" and "CIE" (CIE L*). NULL keeps the actual setting. <code>gammatype="histogram"</code> forces histogram equalization based on the corresponding greyvalue image.
<code>cspace</code>	defines the output color space, default "sRGB" (sRGB D65), alternatives are "Adobe" (Adobe 1998 D65), "wGamut" (Wide Gamut D65), "kodak" (Kodak ProPhoto D65) "xyz", "yuv", "yiq" and "hsi". NULL keeps the actual setting.
<code>whitep</code>	White point in xyY space. Can be given as one of (character) c("A", "B", "C", "E", "D50", "D55", ...) or as a two element numeric vector of chromatic xy coordinates. "D65" corresponds to the default white point of "sRGB" and "Adobe" RGB-spaces. NULL keeps the actual setting.
<code>temp</code>	Color temperature. Can be used to specify chromatic xy coordinates of the whitepoint. Only used if <code>is.null(whitep)</code> .
<code>black</code>	Adjustment for black color. Color values with luminance \leq black will be assigned to black in RGB. Adjustment ist done in xyY space.
<code>exposure</code>	Multiplicative factor for all color channels (in xyz or rgb spaces). Applied in linear color space, i.e. if the image is gamma corrected the gamma correction is reversed first.
<code>...</code>	not used

Details

This functions shows information on the image. This includes histograms of color values in each channel of the specified (`x$type`) color space, a thumbnail (in "sRGB" with gamma correction `gammatype` if specified and `gammatype="ITU"` elsewhere), some information on the image and, if `x` was produced by `awsimage`, an image illustrating the local adaptation.

Value

nothing is returned.

Author(s)

Karsten Tabelow [⟨tabelow@wias-berlin.de⟩](mailto:tabelow@wias-berlin.de) and Joerg Polzehl [⟨polzehl@wias-berlin.de⟩](mailto:polzehl@wias-berlin.de)

See Also

[summary.adimpro](#), [adjust.image](#)

Examples

```
## Not run: demo(color)
```

rotate.image *Image Processing*

Description

Rotate an image by 0, 90, 180 or 270 degrees.

Usage

```
rotate.image(img, angle = 90, compress=NULL)
```

Arguments

img	image data, that is an object of class "adimpro".
angle	0, 90, 180 or 270 degrees
compress	store result as compressed image if compress=TRUE. compress=NULL uses the same format as the original object.

Details

The function rotates the image `img` by 0, 90, 180 or 270 degrees. Any other value for degree will cause an exception. The returned object contains an additional component `rotate` determining the rotation.

Value

An object of class "adimpro" containing the rotated image.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

[show.image](#)

Examples

```
## Not run: demo(manipulate)
```

segment	<i>Segmentation by AWS</i>
---------	----------------------------

Description

The function allows to segment an image into two or three level sets.

Usage

```
segment(object, level=0.5, delta = 0, thresh = 3, fov = NULL, channel = 0, hmax = 4
```

Arguments

object	Image object, class "adimpro", as from <code>read.image</code> , <code>read.raw</code> , or <code>make.image</code> .
level	center of gray/color-values of the second segment, will not be used if <code>select=TRUE</code> . May be specified such that either <code>level-delta</code> and <code>level+delta</code> are within the interval (0,1) or such that they are within the interval (0,65535) (2 Byte integers).
delta	half width of gray/color-values of the second segment, may be increased if <code>select=TRUE</code> . May be specified such that either <code>level-delta</code> and <code>level+delta</code> are within the interval (0,1) or such that they are within the interval (0,65535) (2 Byte integers).
thresh	Critical value for final assignment to segment 1 or 3, should be specified as a quantile of the standard Gaussian distribution.
fov	size of field of view in pixel
channel	specifies which information to use for segmentation. 0: use grey valued image obtained from color images, 1-3: use the specified color channel.
hmax	Maximum bandwidth to use in the iteration procedure.
aws	(logical). If <code>TRUE</code> the propagation - separation (PS) approach from Polzehl and Spokoiny (2006) is used. <code>aws=FALSE</code> turns off the statistical penalty resulting in a nonadaptive kernel estimate using a kernel with bandwidth <code>hmax</code> .
varmodel	<code>varmodel</code> specifies how variances are to be estimated. This can be a homogeneous variance estimate (<code>varmodel="None"</code>) assuming uncorrelated errors (both spatial and between channels). Alternatives are an adaptive homogeneous or linear (function of the mean) variance estimate that depends on estimated correlations and on residuals from the last iteration step. The default <code>varmodel=NULL</code> corresponds to <code>varmodel == "Linear"</code> if <code>img\$gamma==FALSE</code> and <code>varmodel == "Constant"</code> otherwise.
ladjust	adjustment factor for lambda (≥ 1). Default values for lambda are selected for Gaussian distributions. Skewed or heavy tailed distributions may require slightly larger values for lambda to meet the propagation condition. <code>ladjust</code> allows to increase lambda in such situations.
xind, yind	Restrict smoothing to rectangular area defined by pixel indices <code>xind</code> , <code>yind</code> in x- and y-direction. Full range if <code>NULL</code> (default).

<code>wghts</code>	allows to weight the information from different (up to 4) color channels. The weights are used in the statistical penalty of the PS-procedure.
<code>scorr</code>	(logical). Specifies whether spatial correlation is to be estimated. Defaults to <code>TRUE</code> . Is set to <code>FALSE</code> if <code>mask</code> is not <code>NULL</code> .
<code>lkern</code>	Specifies the location kernel. Defaults to "Triangle", other choices are "Quadratic", "Cubic" and "Uniform". The use of "Triangle" corresponds to the Epanechnikov kernel nonparametric kernel regression.
<code>plateau</code>	Extension of the plateau in the statistical kernel. Can take values from (0,1), defaults to 0.25.
<code>homogen</code>	If <code>TRUE</code> the algorithm determines, in each design point i , a circle of maximum radius, such that the statistical penalty $s_{\{ij\}}$ for all points j within the circle is less than the value specified in <code>plateau</code> . In subsequent iteration steps the statistical penalty for such points is set to zero. This is only used if <code>plateau</code> >0. This results in more stable intermediate estimates and in a smoother reconstruction. <code>homogen=TRUE</code> leads to increased memory requirements.
<code>earlystop</code>	If <code>TRUE</code> the algorithm determines, in each design point i , a circle of minimal radius, such that the circle includes all point j with positive weights $w_{\{ij\}}$. If this radius is considerably smaller than the actual bandwidth then the estimate in point i is fixed. This should considerably reduce computing time in case of large <code>hmax</code> . <code>earlystop=TRUE</code> slightly increases memory requirements.
<code>demo</code>	(logical). If <code>demo=TRUE</code> the function pauses after each iteration. Defaults to <code>FALSE</code> .
<code>select</code>	if <code>TRUE</code> a homogeneous rectangular region can be specified interactively. A value of <code>level</code> is generated as the median of values within the selected region.
<code>sext</code>	if <code>select==TRUE</code> the value of <code>delta</code> is increased by <code>sext</code> times the standard deviation (estimated by IQR) of the values in the selected region.
<code>connected</code>	if <code>TRUE</code> the set of pixel within the same segment connected to the specified pixel is extracted.
<code>graph</code>	(logical). If <code>graph=TRUE</code> intermediate results are illustrated after each iteration step. Defaults to <code>FALSE</code> .
<code>max.pixel</code>	Maximum dimension of images for display if <code>graph=TRUE</code> . If the true dimension is larger, the images are downscaled for display. See also show.image .
<code>compress</code>	logical, determines if image data are stored in raw-format.

Details

The image is segmented into three parts by performing multiscale tests of the hypotheses $H1$ value $\geq level - delta$ and $H2$ value $\leq level + delta$. Pixel where the first hypothesis is rejected are classified as -1 (segment 1) while rejection of $H2$ results in classification 1 (segment 3). Pixel where neither $H1$ or $H2$ are rejected are assigned to a value 0 (segment 2). Critical values for the tests are adjusted for smoothness at the different scales inspected in the iteration process using results from multiscale testing, see e.g. Duembgen and Spokoiny (2001). Critical values also depend on the size of the region of interest specified in parameter `fov`.

Within segment 2 structural adaptive smoothing is performed while if a pair of pixel belongs to segment 1 or segment 3 the corresponding weight will be nonadaptive.

If `connected==TRUE` pixel in segment 2 0 are reassigned to a value 2 if they belong to a maximal connected subset of segment2 that contains the center of the specified homogeneous set.

Value

Object of class "adimpro" with

<code>img</code>	containing a greyvalued image with 3 or 4 levels corresponding to the identified segments.
<code>hsegm</code>	containing the maximal bandwidth used
<code>level</code>	the value of parameter <code>level</code> used
<code>delta</code>	the value of parameter <code>delta</code> used
<code>thresh</code>	the value of parameter <code>thresh</code> used

Note

This function is still experimental and may be changes considerably in future.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

References

Duembgen, L. and Spokoiny, V. (2001). Multiscale testing of qualitative hypotheses. *Ann. Stat.* 29, 124–152.

Polzehl, J. and Spokoiny, V. (2006). Propagation-Separation Approach for Local Likelihood Estimation. *Probability Theory and Related Fields.* 3 (135) 335 - 362.

See Also

[read.image](#), [read.raw](#), [make.image](#), [show.image](#), [clip.image](#)

Examples

```
## Not run: demo(segment)
```

show.image

*I/O functions***Description**

Display an image on the screen.

Usage

```
show.image(img, max.x = 1000, max.y = 1000, gammatype = "ITU",
           whitep = NULL, temp = NULL, cspace = "sRGB", black=0, exposure = 1,
           channel=NULL, new = FALSE, ...)
```

Arguments

img	image data, an object of class "adimpro".
max.x	maximum value of pixels in x dimension to be displayed.
max.y	maximum value of pixels in y dimension to be displayed.
gammatype	character, determines the type of gamma correction within the image. "ITU" stands for ITU-R BT.709-3 as e.g. used by ddraw. Alternatives recognized within the package are "None", "sRGB" and "CIE" (CIE L*). gammatype="histogram" forces histogram equalization based on the corresponding greyvalue image.
cspace	defines the output color space, default "sRGB" (sRGB D65), alternatives are "Adobe" (Adobe 1998 D65), "wGamut" (Wide Gamut D65), "kodak" (Kodak ProPhoto D65) "xyz", "yuv", "yiq" and "hsi". NULL keeps the actual setting. If <code>color.space%in%c("hsi", "yuv", "yiq", "xyz")</code> the individual channels are rescaled to provide maximum contrast. Information from the three channels is coded as "red", "green" and "blue" providing a miscolored image in these cases. Individual channels can be displayed as greyvalue images by specifying the channel argument. <code>color.space="greyvalue"</code> provides a greyvalue image.
whitep	White point in xyY space. Can be given as one of (character) c("A", "B", "C", "E", "D50", "D55", ...) or as a two element numeric vector of chromatic xy coordinates. "D65" corresponds to the default white point of "sRGB" and "Adobe" RGB-spaces. NULL keeps the actual setting.
temp	Color temperature. Can be used to specify chromatic xy coordinates of the whitepoint. Only used if <code>is.null(whitep)</code> .
black	Adjustment for black color. Color values with luminance \leq black will be assigned to black in RGB. Adjustment ist done in xyY space.
exposure	Multiplicative factor for all color channels (in xyz or rgb spaces). Applied in linear color space, i.e. if the image is gamma corrected the gamma correction is reversed first.
channel	allows to select a color channel (1: red, 2: green, 3: blue in case of "rgb") for display.
new	should new X11() be opened? default FALSE
...	additional arguments to image can be passed here.

Details

This function displays greyscale and color images on the screen. If the actual dimension of the image exceeds `max.x` or `max.y` the image is shrunk by displaying only part of the pixels (every second/third/... such that the resulting dimension is smaller than `max.x` or `max.y`)

Value

Nothing is returned

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

[read.image](#), [write.image](#), [adjust.image](#)

Examples

```
## Not run: demo(io)
```

shrink.image *Image Processing*

Description

Shrink resolution of an image

Usage

```
shrink.image(img, method = "median",  
             xt = img$dim[1], yt = img$dim[2], ratio = TRUE,  
             compress=TRUE)
```

Arguments

<code>img</code>	image data, an object of class "adimpro".
<code>method</code>	method to be used to shrink the image. "median" (default), "mean", or "nearest". "median" is supposed to give best results. For a considerably faster result use "nearest".
<code>xt</code>	target x-dimension
<code>yt</code>	target y-dimension
<code>ratio</code>	logical. preserve x-y ratio? default: TRUE
<code>compress</code>	logical, determines if image data are stored in raw-format.

Details

This function shrinks the resolution of the image such that the x-y dimension of the resulting image is smaller than the original one. `xt` and `yt` give the target dimension of the image. If `ratio == TRUE` (default) the ratio between x- and y-dimension is preserved.

method "nearest" selects a pixel nearest to the new coordinates. method "mean" defines the color of a pixel as the mean of all pixel identified with the new coordinate. method "median" set the color of a pixel as color of an L1-generalized median of all pixel identified with the new coordinate.

Value

shrunked image

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

Examples

```
## Not run: demo(manipulate)
```

summary.adimpro *I/O functions*

Description

'summary' method for class '"adimpro"'.

Usage

```
## S3 method for class 'adimpro':
summary(object, ...)
```

Arguments

`object` an object of class `adimpro`, usually, a result of a call to `read.raw`, or `read.image`.
`...` further arguments passed to or from other methods.

Details

The method tries to print information on the image, like image dimension, color space, value range, etc.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

Examples

```
## Not run: demo(io)
## Not run: demo(manipulate)
```

write.image	<i>I/O Functions</i>
-------------	----------------------

Description

Write an image file.

Usage

```
write.image(img, file = "tmp.ppm", max.x=NULL, max.y=NULL, depth = NULL, gammatype=
           whitep = NULL, temp = NULL, cspace = NULL, black = 0, exposure = 1)
```

Arguments

img	image data, an object of class "adimpro".
file	file name, the extension determines the image file format.
max.x	maximum value of pixels in x dimension.
max.y	maximum value of pixels in y dimension.
depth	color depth, either 8 or 16 (bit)! The value is reset to 8 if the specified image file format does not allow for 16 Bit encoding.
gammatype	character, determines the type of gamma correction within the image. "ITU" stands for ITU-R BT.709-3 as e.g. used by ddraw. Alternatives recognized within the package are "None", "sRGB" and "CIE" (CIE L*). NULL keeps the actual setting. gammatype="histogram" forces histogram equalization based on the corresponding greyscale image.
cspace	defines the output color space, default is img\$cspace can be set to "sRGB", "Adobe" (Adobe 1998), "wGamut" (Wide Gamut), "kodak" (Kodak ProPhoto) "xyz", "yuv", "yiq" and "hsi"
whitep	White point in xyY space. Can be given as one of (character) c("A", "B", "C", "E", "D50", "D55", "D65", "D68", "D94", "D98", "E", "F", "M", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z") or as a two element numeric vector of chromatic xy coordinates. "D65" corresponds to the default white point of "sRGB" and "Adobe" RGB-spaces. NULL keeps the actual setting.
temp	Color temperature. Can be used to specify chromatic xy coordinates of the whitepoint. Only used if is.null(whitep).
black	Adjustment for black color. Color values with luminance <= black will be assigned to black in RGB. Adjustment ist done in xyY space.
exposure	Multiplicative factor for all color channels (in xyz or rgb spaces). Applied in linear color space, i.e. if the image is gamma corrected the gamma correction is reversed first.

Details

This function writes the image data in `img` to the file `file`. Color depth `depth` is used for writing, but if image has a component "depth", this argument will be ignored. Note: Not all target formats support 16bit coding. The target format is determined from the file extension, and should be one of the many that ImageMagick supports.

Note that `write image` by default applies a gamma correction with `gammatype="ITU"`. This provides a good standard for display on a screen. For printing `cspace="Adobe"` should be preferred. Images that are intended for further editing should preferably be saved in an image format that allows for 16Bit depth (tiff, png) using either `cspace="Adobe"`, `cspace="wGamut"` or `cspace="kodak"` and preferably no gamma correction to prevent from additional loss of information.

Value

Nothing is returned.

Note

ImageMagick has to be installed on the system to write "tif", "tiff", "png", "gif", "jpg" or "jpeg" files.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

[read.image](#), [adjust.image](#)

Examples

```
## Not run: demo(io)
```

`write.raw`

Write image RAW data as greyscale png image

Description

Image RAW data is saved as a 16-Bit greyscale png image. EXIF information contained in the original RAW image is stored as a comment.

Usage

```
write.raw(img, filename = "tmp.png")
```

Arguments

`img` object of class "adimpro" containing image RAW data (`img$type=="RAW".`)
`filename` Name of the resulting png-image. If filename does not include an extension ".png" the extension ".png" is added.

Details

EXIF information contained in the original RAW image as well as other available information in object `img` are added as a comment to the resulting png-image. This comment is evaluated when the image is read by functions [read.raw](#) or [read.image](#)

Value

Nothing is returned.

Note

ImageMagick has to be installed on the system to write "png" files.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de> and Joerg Polzehl <polzehl@wias-berlin.de>

See Also

[read.raw](#), [develop.raw](#)

Examples

```
## Not run: demo(raw)
```

Index

*Topic **IO**

- adimpro, 2
- develop.raw, 15
- extract.info, 18
- write.image, 30
- write.raw, 31

*Topic **color**

- colorspace, 12

*Topic **hplot**

- plot.adimpro, 21
- show.image, 27

*Topic **iplot**

- plot.adimpro, 21

*Topic **manip**

- adjust.image, 4
- awsimage, 6
- awsraw, 9
- clip.image, 11
- combine, 13
- edges, 16
- extract.image, 17
- extract.ni, 19
- imganiso2D, 19
- mask.create, 20
- rotate.image, 23
- shrink.image, 28

*Topic **print**

- summary.adimpro, 29

*Topic **smooth**

- awsimage, 6
- awsraw, 9
- segment, 24

*Topic **utilities**

- adimpro, 2
- adimpro.options, 4
- develop.raw, 15
- extract.info, 18
- write.image, 30
- write.raw, 31

- adimpro, 2

- adimpro.options, 4

- adjust.image, 4, 22, 28, 31

- awsaniso, 20

- awsaniso (*awsimage*), 6

- awsimage, 6, 10, 11, 19, 21

- awspimage, 11, 19

- awspimage (*awsimage*), 6

- awsraw, 9

- clip.image, 8–10, 11, 18, 20, 26

- colorspace, 12

- combine, 13

- develop.raw, 10, 15, 18, 32

- edges, 16

- extract.image, 17

- extract.info, 18

- extract.ni, 19

- hsi2rgb (*colorspace*), 12

- image, 27

- imganiso2D, 19

- make.image, 9–11, 14, 17, 20, 26

- make.image (*adimpro*), 2

- mask.create, 20

- plot.adimpro, 21

- read.image, 3, 9, 11, 15, 18, 20, 26, 28, 31, 32

- read.image (*adimpro*), 2

- read.raw, 3, 9–11, 15, 18, 20, 26, 32

- read.raw (*adimpro*), 2

- rgb2grey (*colorspace*), 12

- rgb2hsi (*colorspace*), 12

- rgb2xyz (*colorspace*), 12

- rgb2yiq (*colorspace*), 12

`rgb2yuv` (*colorspace*), 12
`rotate.image`, 23

`segment`, 24
`show.image`, 6, 8–11, 19, 20, 23, 25, 26, 27
`shrink.image`, 28
`summary`, 18
`summary.adimpro`, 22, 29

`write.image`, 6, 18, 19, 28, 30
`write.raw`, 18, 31

`xyz2rgb` (*colorspace*), 12

`yiq2rgb` (*colorspace*), 12
`yuv2rgb` (*colorspace*), 12