Computer Vision mit R

# 1. Computer Vision mit R

# Vorbereitungen

## Clear workspace

Löschen Sie die alte Umgebung, um in einem “neuen” R zu starten.

rm(list=ls())

# 2. Bildverarbeitung in R mit Imager

## Installation von imager

if (!require("imager")) install.packages("imager")

## Lade nötiges Paket: imager

## Warning: Paket 'imager' wurde unter R Version 4.4.3 erstellt

## Lade nötiges Paket: magrittr

##   
## Attache Paket: 'imager'

## Das folgende Objekt ist maskiert 'package:magrittr':  
##   
## add

## Die folgenden Objekte sind maskiert von 'package:stats':  
##   
## convolve, spectrum

## Das folgende Objekt ist maskiert 'package:graphics':  
##   
## frame

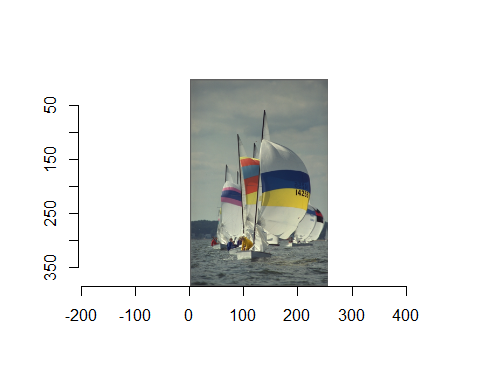
## Das folgende Objekt ist maskiert 'package:base':  
##   
## save.image

library(imager)

## Bilder laden und plotten

In der imager Bibliothek können darin enthaltene Bilder geladen und angezeigt werden.

library(imager)  
plot(boats)

 Die Klasse von Bilder ist “cimg” und es können Basisinformationen zum Bild abgerufen werden.

class(boats)

## [1] "cimg" "imager\_array" "numeric"

boats

## Image. Width: 256 pix Height: 384 pix Depth: 1 Colour channels: 3

Das Bild enhält gemäß RGB drei Kanäle, eine Graustufenbild hätte nur einen.

grayscale(boats)

## Image. Width: 256 pix Height: 384 pix Depth: 1 Colour channels: 1

Sie können die Dimensionen von Bilddaten untersuchen

dim(boats)

## [1] 256 384 1 3

Mit Bildern können, da es sich um Arrays handelt, also mehrdimensionale Matrizen, deren Werte den Farbintensitätswerten der verschiedenen Kanäle entsprechen, können arithmetische Operationen durchgeführt werden.

log(boats)+3\*sqrt(boats)

## Image. Width: 256 pix Height: 384 pix Depth: 1 Colour channels: 3

mean(boats)

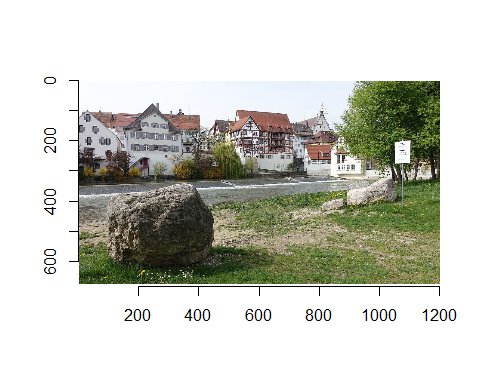
## [1] 0.5089061

sd(boats)

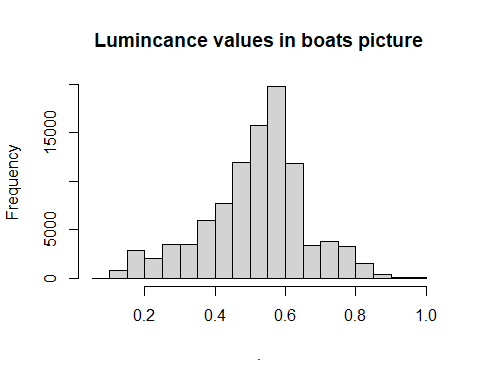
## [1] 0.144797

Sie können Bilder auch außerhalb der Bibliothek laden, wenn Sie den Pfad des Bildes angeben, das Sie laden wollen.

exampleimage <- load.image("https://upload.wikimedia.org/wikipedia/commons/thumb/5/55/Riedlingen\_-\_Stadt\_%C3%BCber\_der\_Donau.JPG/1200px-Riedlingen\_-\_Stadt\_%C3%BCber\_der\_Donau.JPG")  
plot(exampleimage)

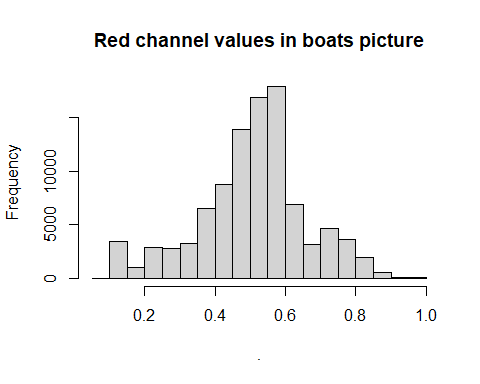
 ## Histogramme der Pixelwerte Sie können Histogramme aus Bilddaten erzeugen, um die Verteilung der Pixelwerte zu untersuchen

grayscale(boats) %>% hist(main="Lumincance values in boats picture")



Wenn nur ein ein Farbkanal untersucht werden soll, können Sie für den roten Farbkanal folgendes tun:

R(boats) %>% hist(main="Red channel values in boats picture")

 Sie können Bilder auch in einen Dataframe überführen und die Farbkanäle gleichzeitig darstellen:

library(ggplot2)

## Warning: Paket 'ggplot2' wurde unter R Version 4.4.3 erstellt

library(dplyr)

## Warning: Paket 'dplyr' wurde unter R Version 4.4.3 erstellt

##   
## Attache Paket: 'dplyr'

## Das folgende Objekt ist maskiert 'package:imager':  
##   
## where

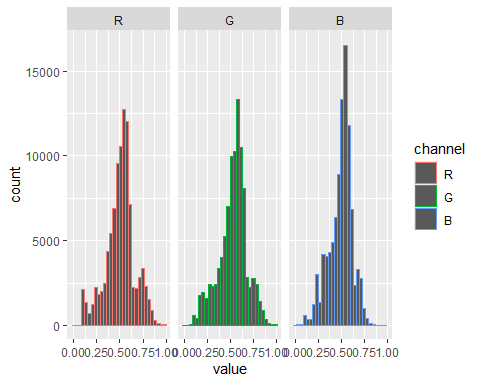
## Die folgenden Objekte sind maskiert von 'package:stats':  
##   
## filter, lag

## Die folgenden Objekte sind maskiert von 'package:base':  
##   
## intersect, setdiff, setequal, union

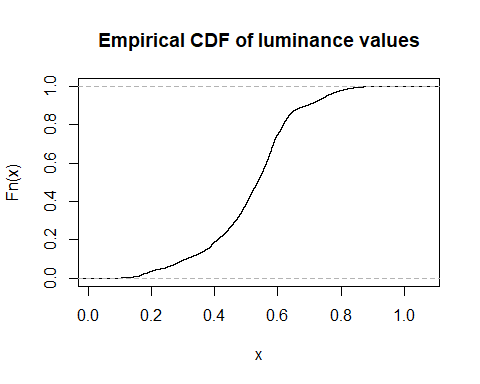
bdf <- as.data.frame(boats)  
head(bdf,3)

## x y cc value  
## 1 1 1 1 0.3882353  
## 2 2 1 1 0.3858633  
## 3 3 1 1 0.3849406

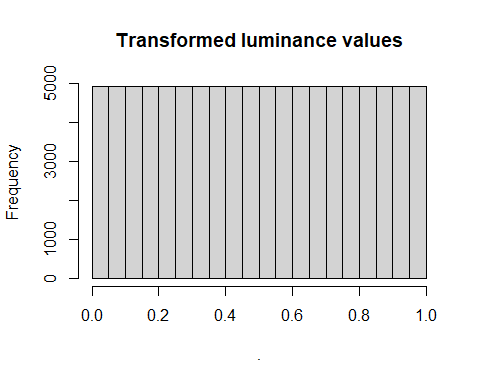
bdf <- mutate(bdf,channel=factor(cc,labels=c('R','G','B')))  
ggplot(bdf,aes(value,col=channel))+geom\_histogram(bins=30)+facet\_wrap(~ channel)

 Sie können eine Histogram Equalization benutzen um Kontrastunterschiede im Bild auszugleichen und so eine gleichmäßige Verteilung erreichen:

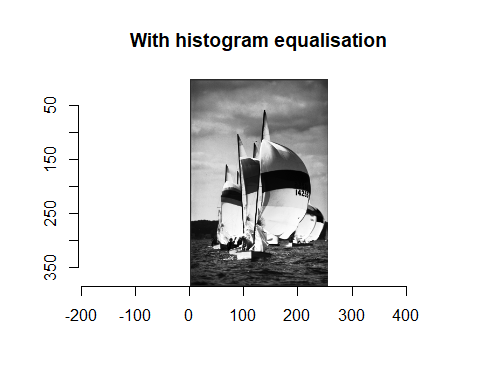
boats.g <- grayscale(boats)  
f <- ecdf(boats.g)  
plot(f,main="Empirical CDF of luminance values")



f(boats.g) %>% hist(main="Transformed luminance values")

 Um das Bild darzustellen, müssen Sie zunächst den Datentyp ändern:

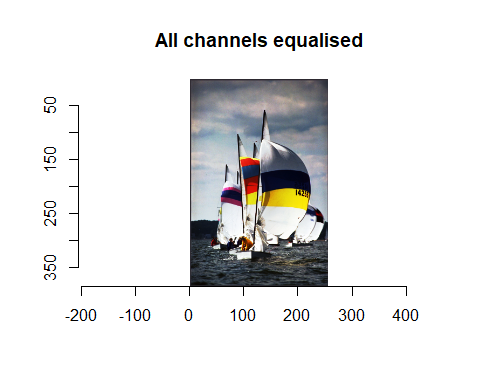
f(boats.g) %>% as.cimg(dim=dim(boats.g)) %>% plot(main="With histogram equalisation")

 Dies kann ebenso mit RGB Bildern durchgeführt werden:

#Hist. equalisation for grayscale  
hist.eq <- function(im) as.cimg(ecdf(im)(im),dim=dim(im))  
  
#Split across colour channels,  
cn <- imsplit(boats,"c")  
cn #we now have a list of images

## Image list of size 3

cn.eq <- map\_il(cn,hist.eq) #run hist.eq on each  
imappend(cn.eq,"c") %>% plot(main="All channels equalised") #recombine and plot



Führen Sie nun eine Histogram Equalization mit dem Bild von Riedlingen durch.

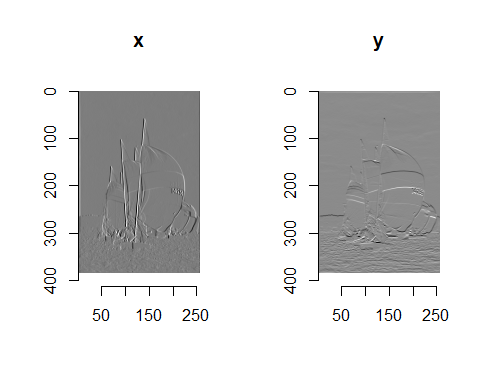
## Edge Detection

Mit Image Gradients kann eine Edge Detection durchgeführt werden:

gr <- imgradient(boats.g,"xy")  
gr

## Image list of size 2

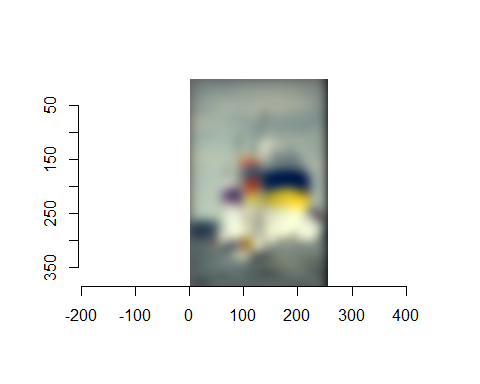
plot(gr,layout="row")

 Die beiden dargestellten Komponenten entsprechen der x- und der y-Richtung der Gradienten. Mathematisch ausgedrückt wird eine Edge mit einem abrupten Wechsel im Bild erkannt, also der Änderung der Bildintensität.

Führen Sie auch die Edge Detection mit dem Riedlingen Bild durch

## Blur

im.blurry <- isoblur(boats,10) #Blurry parrots!  
plot(im.blurry)



## Save images

imager::save.image(boats,"boats.jpg")