

Bildbearbeitung im Terminal mit Imagemagick

Stand: 09.05.2024

Quellen: <https://www.freiesoftwareog.org>, Rene Schwarzinger



Schnell mal 800 Bilder umbenennen, skalieren und durchnummerieren -

Kein Problem im Terminal...

- Edgar „Fast Edi“ Hoffmann, Pirat und Freie-Software-Aktivist

Inhaltsverzeichnis

Allgemeines.....	3
Installation.....	4
Typische Anwendungsbereiche.....	4
Wichtiger Hinweis.....	4
Praktische Anwendung und Beispiele.....	5
Betrachten, Übersichtsmodus und Informationsgewinnung.....	5
Konvertieren, Skalieren und Umbenennen mit „convert“.....	5
Skalieren und Drehen mit „mogrify“.....	6
Rahmen, Schatten und Tiefe erzeugen.....	6
Einfache Rahmenlinie.....	6
Rahmen aus Bild.....	6
Anspruchsvolle Rahmen.....	7
Schatten.....	7
Tiefe.....	7
Weicher, ovaler Rahmen.....	7
Weicher, nicht rein ovaler Rahmen.....	7
„Runde“ Ecken.....	7
Polaroid.....	8
Screenshots erzeugen.....	8
Thumbnails.....	8
Variante mit eigener Beschriftung.....	8
HTML Thumbnails.....	9
HTML Thumbnails mit Polaroid Effekt - 1 (komplex, ohne exakte Erklärung).....	9
HTML Thumbnails mit Polaroid Effekt - 2 (komplex, ohne exakte Erklärung).....	9
Alternative mittels convert und visual directories.....	9
Filter.....	9
Scharfzeichnen.....	9
Aufhellen.....	9
Graustufen bzw. S/W.....	9
Kohlezeichnung.....	10
Negativ.....	10
Strudel.....	10
Implodieren.....	10
Wasserzeichen erzeugen.....	11
Beschriftung.....	13
Text mit Schatten Variante 1.....	13
Text mit Schatten Variante 2.....	13
Text mit einfacher Rahmenlinie.....	14
Text mit doppelter Rahmenlinie.....	14
Durchsichtiger Text.....	14
Text mit „Mahjongg-Effekt“.....	14
Weitere Informations- und Ideenquellen.....	15
Die Communities.....	15
Internetquellen.....	15
Schlussbetrachtung.....	15
Eigene Notizen.....	16

Allgemeines

Wenn es um effektive Vorgänge geht, ist GNU/Linux der absolute King. Nicht umsonst gilt hier die Prämisse „One Job, one tool“.

ImageMagick ist ein freies, quelloffenes Softwarepaket zur Erstellung und Bearbeitung von Rastergrafiken.

Es kann momentan mehr als 200 der meist verwendeten Bildformate lesen, verändern und schreiben.

Außerdem lassen sich Bilder dynamisch generieren, weshalb es auch von Webanwendungen verwendet wird.

ImageMagick ist kein Einzelprogramm, sondern ein Paket, bestehend aus 11 Kommandozeilenprogrammen, die alle auf einen gemeinsamen Satz von Bibliotheken zugreifen, die wiederum das Schreiben und Lesen vieler verschiedener Dateiformate und umfangreiche grafische Arbeiten ermöglichen:

- 🐞 **animate** - spielt mehrere Bilder schnell hintereinander ab
- 🐞 **convert** - liest Bilder, bearbeitet sie und speichert sie ab
- 🐞 **compare** - vergleicht 2 Bilder und gibt die Unterschiede als Bilddatei aus
- 🐞 **composite** - überlagert mehrere Bilder zu einem Bild
- 🐞 **conjure** - führt Skripte in der Skriptsprache von ImageMagick aus
- 🐞 **display** - stellt Bilder auf dem Bildschirm dar
- 🐞 **identify** - gibt Dateiformat, Bildgröße usw. von Bilddateien aus
- 🐞 **import** - macht Bildschirmfotos
- 🐞 **montage** - fasst mehrere Bilder zu einem großen Einzelbild zusammen
- 🐞 **mogrify** - wie convert, nur dass die Eingabedatei durch die Ausgabedatei ersetzt wird!
- 🐞 **stream** - liest aus Bilddateien Teile aus und gibt sie als Rohdaten, Fließkommazahlen oder ähnliches aus

Einige dieser Werkzeuge werden in dieser Broschüre anhand praktischer Beispiele vorgestellt. Wir können natürlich nicht alle phantastischen Möglichkeiten dieser Sammlung aufgreifen. Hier empfiehlt sich eine Internet-Suche.

Hinweis:

Zu jedem oben genannten Kommandozeilenprogramm gibt es eine Anleitung (sogenannte „manpage“).

Aufgrund der vielen möglichen Parameter und Optionsschalter (ca. 241) ist es ratsam, auch die Dokumentation zu installieren. Diese findet sich dann unter:

/usr/share/doc/imagemagick-doc/index.html

und

/usr/share/doc/imagemagick-doc/www/command-line-options.html

Installation

In einigen GNU/Linux-Distributionen wird ImageMagick vorinstalliert mitgeliefert, bei anderen nicht. Zum Testen kann man im Terminal beispielsweise versuchen, eines der Werkzeuge aufzurufen. Erscheint hier ein Fehler, muss man das Paket nachinstallieren:









```
convert --version
```

In jedem Fall findet es sich in den offiziellen Repositories der meisten GNU/Linux-Distributionen und kann im Terminal ganz einfach installiert werden:

```
sudo apt install imagemagick imagemagick-doc
```

Typische Anwendungsbereiche

Die ImageMagick-Programme eignen sich besonders für die folgenden Aufgabenbereiche:

-  Konvertieren in andere Bildformate
-  Ändern der Bildgröße
-  Pixelgenau drehen, skalieren und ausschneiden
-  Anwenden von Filtern und Effekten
-  Erstellung von Animationen für das Web
-  Erstellen von einfachen Grafiken mittels geometrischer Grundformen oder durch Kopieren von (kleineren) Grafikschnipseln
-  Erzeugen von Thumbnails und Rahmen
-  Mehrere Bilder zu einem Gesamtwerk zusammenstellen (etwa als Poster, Collage oder Bildergalerie)

Wichtiger Hinweis

Wie bei allen Dateioperationen (auch grafischen Lösungen wie Gimp) kann bei der Verwendung der ImageMagick Werkzeuge auch immer etwas schief gehen.

Der falsche Dateiname, ein verkehrter Parameter, etc...

Deshalb ist es gute Praxis, Bildmanipulationen nur an Kopien der Originalbilder durchzuführen!

Praktische Anwendung und Beispiele

In diesem Kapitel geht es darum, praktische Anwendungsfälle in aller Kürze aufzuzeigen und mit dem entsprechenden Hilfsprogramm mehr oder weniger komplex umzusetzen.

Betrachten, Übersichtsmodus und Informationsgewinnung

Zum Betrachten aller .jpg Bilder in einem Ordner dient das Kommando

```
display *.jpg
```

Mit der Leertaste blättert man zum nächsten Bild.

Ein Menü kann über die linke Maustaste aufgerufen werden.

Eine Übersicht über den Fotobestand, liefert der Befehl

```
display vid:*.jpg
```

Diese Übersicht kann man via Strg+S auch speichern.

Details zu einem Bild werden über folgende Anweisung abgerufen:

```
identify bild.jpg
```

Mit dem Parameter *-verbose* wird die Ausgabe wesentlich detaillierter.

Daher empfiehlt es sich, zu „pipen“ und zu filtern:

```
identify -verbose bild.png | grep -A5 „Properties:“
```

Dies gibt die Infos unter der Zeichenkette „Properties:“ aus (5 Zeilen).

Ein weiteres Beispiel:

```
identify -verbose bild.png | grep -A13 „Image:“
```

Konvertieren, Skalieren und Umbenennen mit „convert“

Es ist sehr einfach, ein Bild von einem Format in ein anderes zu konvertieren:

```
convert bild.jpg bild.png
```

Will man zum Beispiel alle .webp-Bilder in .png umwandeln, geht das sehr einfach:

```
for BildDatei in *.webp; do convert ./"$BildDatei" ./"${BildDatei%.webp}.png"; done
```

Genauso verhält es sich mit dem Skalieren:

```
convert Bild_gross.jpg -resize 640x480 Bild_klein.jpg
```

Anstatt der pixelgenauen Auflösung kann auch eine Angabe in % erfolgen.

Ebenso kann an Stelle von *-resize* auch *-sample*, *-geometry* oder *-thumbnail* verwendet werden.

Genauso ist eine Angabe von „640x“ bzw. „x480“ möglich.

ImageMagick achtet bei allen Befehlen auf das richtige Seitenverhältnis beim Skalieren. Wenn dieses Verhalten geändert werden soll, muss ein Ausrufezeichen nach der Größenangabe folgen: z.B: 200x150!

Eine weitere, oft auftauchende Aufgabe ist das Durchnummerieren von Bildern. Dies lässt sich sehr schnell mit einem Skript...

```
let a=0
for i in *.png; do
    let a=a+1;
    convert $i name_$.png
done
```

...oder als einzelner Terminal-Befehl erledigen.

```
let a=0; for i in *.png; do; let a=a+1; convert $1 name_$.png; done
```

Als kleines „Sahnehäubchen“ kann man die drei Aufgaben auch in einem erledigen:

```
let a=0
for i in *.jpg; do
    let a=a+1;
    convert $i -resize 640x480 ./neu/name_$.png
done
```

Skalieren und Drehen mit „mogrify“

Zu beachten ist hierbei, dass der Befehl mogrify im Gegensatz zu convert das Originalbild manipuliert! Deshalb an dieser Stelle nochmals der Hinweis, immer mit Kopien der Originalbilder zu arbeiten, damit bei Fehleingaben oder Programmfehlern kein Schaden entsteht.

Eine weitere Möglichkeit der Skalierung wäre wie folgt:

```
mogrify -format png -resize 25% *.jpg
```

Hierbei wandelt man alle jpg-Dateien des Verzeichnisses in png um und skaliert diese gleichzeitig auf 25% der Originalgröße.

Möchte man zum Beispiel Bilder drehen (vom Hoch- zum Querformat oder umgekehrt), geht das sehr einfach:

```
mogrify -rotate „90>“ bild.png # 90° nach rechts
mogrify -rotate „90<“ bild.png # 90° nach links
```

Rahmen, Schatten und Tiefe erzeugen

Einfache Rahmenlinie

Folgende Anweisung bewirkt eine dünne, schwarze Rahmenlinie um das Bild.

```
mogrify -bordercolor "#ff0000" -border 1x1 bild.png
```

Rahmen aus Bild

Durch unterschiedlich farbige Ränder kann ein Bild scheinbar etwas aus der Ebene hervorstehen (*-raise*) bzw. darunterliegen (*+raise*).

```
convert alt.png -raise 15x15 neu.png
convert alt.png +raise 15x15 neu.png
```

Anspruchsvolle Rahmen

Mit der Kombination *-frame* und *-mattecolor* können anspruchsvollere Rahmen erzeugt werden.

```
convert -mattecolor black alt.png -frame 10x10+5 neu.png
convert -mattecolor black alt.png -frame 10x10+5+5 neu.png
convert -mattecolor gray alt.png -frame 15x15+0+15 neu.png
convert -mattecolor gray alt.png -frame 15x15+15+0 neu.png
```

Die Syntax von *-frame* gehorcht dabei folgender Formel:
Rahmenbreite x Rahmenhöhe + äußererSchatten + innererSchatten

Schatten

Der erste Parameter von *-shadow* ist für die Transparenz der Schattenfarbe verantwortlich, der zweite Wert bewirkt einen sanfteren Schatten und die letzten beiden Angaben sind für den XY-Versatz zuständig.

```
convert alt.png +clone -background black -shadow 80x0+5+5 +swap
-background none -mosaic neu.png
```

Tiefe

Diese Methode erzeugt einen 3D Eindruck.

```
convert alt.png +clone -fill DarkSlateGrey -colorize 100% -repage +0+1 +clone
-repage +1+2 +clone -repage +1+3 +clone -repage +2+4 +clone -repage
+2+5 +clone -repage +3+6 -background none -compose DstOver -mosaic
neu.png
```

Weicher, ovaler Rahmen

Interessante Effekte können vor allem durch Verändern des zweiten Parameters von *-vignette* erzielt werden – die vollständige Syntax lautet:

Radius x Sigma ± Versatz X ± Versatz Y

```
convert alt.png -background none -vignette 0x20 neu.png
```

Weicher, nicht rein ovaler Rahmen

Folgende Methode legt transparente Pixeln über die Ränder und Ecken des Bildes. Unterschiedliche Effekte können auch hier am besten durch Verstellen des zweiten Parameters von *-blur* erreicht werden. Interessant ist auch ein Vertauschen der Parameter von *-level*.

```
convert alt.png -alpha set -virtual-pixel transparent -channel A
-blur 0x20 -level 0,50% neu.png
```

„Runde“ Ecken

Das Ergebnis weist in dieser Form leicht ausgefranste Ecken aus:

```
convert alt.png -alpha set -virtual-pixel transparent -channel A
-blur 0x20 -threshold 50% neu.png
```

Polaroid

Die zusätzlich Option *+polaroid* (+ ist Standardeinstellung mit zufälliger Drehung) in Kombination mit *-bordercolor* (Umrandung) und *-background* (Schatten) manipuliert das Bild so, dass es wie ein eingeklebtes, sich leicht wellendes Foto aussieht.

```
mogrify -bordercolor white -background black +polaroid bild.png
```

-polaroid ermöglicht die Angabe des Drehwinkels und *-thumbnail* wirkt wie *-resize*, nur werden hierbei Bildinformationen (z.B. Kommentare, ...) gelöscht.

```
convert -bordercolor AliceBlue -background SteelBlue4 alt.png  
-thumbnail 120x120 -polaroid 5 neu.png
```

Eine Beschriftung kann über *-caption* (Auslesen der Metadaten, Vergabe eines individuellen Titels) erreicht werden.

```
convert -caption „Beschriftung“ -font Courier -pointsize 18 -bordercolor Snow  
-background black -fill dodgerblue -stroke navy -gravity center alt.png  
-thumbnail 120x120 -polaroid 5 neu.png
```

Screenshots erzeugen

Der Befehl *import* stellt ein Fadenkreuz bereit, wodurch ein Bildschirmfoto durch Klicken in ein Fenster erzeugt werden kann – die Speicherung erfolgt im Home-Verzeichnis des aktuellen Benutzers.

```
import bild.png
```

Alternativ kann ein Fenster bzw. der gesamte Desktop gleich als Parameter übergeben werden:

```
import -window terminal bild.png  
import -window root bild.png
```

Thumbnails

Der folgende Befehl legt um jedes Vorschaubild einen Rahmen, schreibt den Dateinamen samt Auflösung darunter und packt schließlich die fertigen Grafiken in eine Datei. Mittels *-tile x{Zahl}* kann die Anzahl der Zeilen festgelegt werden.

```
montage -label '%f\n%wx%h' [-tile x1] [-geometry '200x150+5+5']  
[-frame 5] [-background transparent] *.png uebersicht.png
```

Variante mit eigener Beschriftung

Über das Attribut *-title* kann den Thumbnails eine Überschrift verpasst werden. Mittels *-pointsize* kann die Schriftgröße verändert werden – dies ist vor allem bei langen Dateinamen empfehlenswert. *-shadow* erzeugt einen Schatten rund um die einzelnen Thumbnails.

```
montage -title 'Hier steht ein Text' -pointsize 8 -label Titel1 Datei1.png  
-label Titel2 Datei2.png -labe3 Titel3 Datei1.png -tile x2  
-geometry '200x150+5+5' -frame 5 -background transparent  
-shadow uebersicht.png
```


HTML Thumbnails

Möchte man das Ganze als Webseite, muss nur der entsprechende Dateityp angegeben werden (.html statt .png):

```
montage ... uebersicht.html
```

Hierbei werden drei Dateien erzeugt, wobei die Datei `vid_index_map.shtml` gelöscht werden kann.

HTML Thumbnails mit Polaroid Effekt - 1 (komplex, ohne exakte Erklärung)

-*bordercolor* regelt die Farbe der Umrahmung und das erste -*background* die Farbe des Schattens. Um zwischen den Bildern wieder eine weiße Farbe zu erhalten, muss das Attribut -*background* anschließend zurückgesetzt werden.

```
montage -caption '%t' -bordercolor AliceBlue -background grey20 -geometry +1+1  
-tile 5x './*.png' -thumbnail 128x128 +polaroid -background white \  
polaroid_index.html
```

HTML Thumbnails mit Polaroid Effekt - 2 (komplex, ohne exakte Erklärung)

```
montage -bordercolor Lavender -background black -geometry -10+2  
-tile x1 null: './*.png' null: -thumbnail 200x200 +polaroid -resize 30%  
-background LightGray polaroid2_index.html
```

Alternative mittels convert und visual directories

```
convert 'vid:../fotos/*' vid_index.png
```

oder

```
convert 'vid:../fotos/*' vid_index.html
```

Dabei sollte beachtet werden, dass die Attribute -*tile*, -*frame* und -*shadow* andere Parameter erwarten als dies bei montage der Fall ist.

Filter

Es gibt auch viele Filter, die man auf einfache Art und Weise per Parameter auf ein Bild anwenden kann.

Scharfzeichnen

```
convert alt.png -sharpen 5 neu.png
```

Aufhellen

```
convert alt.png -sigmoidal-contrast 4,0% neu.png
```

Graustufen bzw. S/W

```
convert alt.png -colorspace gray neu.png
```

Achtung: Hier muss das zu bearbeitende Bild zuerst angegeben werden!

```
convert -channel R alt.png -separate neu.png
```

oder

```
convert alt.png -monochrome grau.png
```

Strudel

```
convert alt.png -swirl 90 neu.png
```

-swirl verdreht Pixel um eine Gradangabe um das Zentrum des Bildes.

Implodieren

```
convert alt.png -implode 0.5 neu.png
```

-implode implodiert Pixel um einen Faktor um das Zentrum des Bildes.

Kohlezeichnung









```
convert alt.png -charcoal 2 neu.png
```

Der Wert hinter *-charcoal* bestimmt dabei die Härte des Kohlestiftes.

Negativ

```
convert alt.png -solarize 0% neu.png
```

Hiermit kann ein Negativ wie bei der analogen Fotografie erzeugt werden.

Scharfzeichnen	Aufhellen	Graustufen	Schwarz/Weiss
			
Strudel	Implodieren	Kohlezeichnung	Negativ
			

Wasserzeichen erzeugen

Mit Hilfe von ImageMagick lassen sich Wasserzeichen in Bilder einbringen.

```
composite [-gravity SouthEast] -watermark 30% [-geometry +50+50]  
logo.png alt.png neu.png
```

Der Befehl `composite` kopiert das Bild aus `logo.png` in jene Ecke, die über `-gravity` als Bezugspunkt angegeben wurde (hier: unten links). Mittels `-geometry +50+50` kann zusätzlich eine pixelgenaue Positionierung bezogen auf den jeweiligen Bezugspunkt erreicht werden. Die Prozentangabe nach `-watermark` bewirkt ein Durchscheinen des Wasserzeichens.



Das Ganze kann man bei mehreren zu bearbeitenden Dateien, was bei Fotos durchaus öfter vorkommen könnte, auch sehr gut in ein Skript „giessen“:

```
#!/bin/bash
# Wasserzeichentext in alle JPEG Bilder aus diesen Verzeichnis einfüegen
# Der Wasserzeichentext wird unten links ins Bild eingebracht
# Man kann folgende Parameter anpassen:
Textabstandvonlinks=10
Textabstandvonunten=10
Schriftgroesse=10
# Der Pfad ist je nach Distribution unterschiedlich!
PfadFonts="/usr/share/fonts/truetype/freefont"
Schriftart="FreeSans.ttf"
Schriftfarbe="white"
# Moegliche Farben koennen aufgelistet werden mit dem Befehl:
# convert -list color
Wasserzeichentext="CC-BY-SA FreieSoftwareOG"
# -- Programmbeginn --
echo "Textabstand von links: $Textabstandvonlinks"
echo "Textabstand von unten: $Textabstandvonunten"
echo "Schriftgoesse: $Schriftgroesse"
echo "Schriftart: $Schriftart"
echo "Schriftfarbe: $Schriftfarbe"
echo "Wasserzeichentext: $Wasserzeichentext"
echo " "
for file in *.JPG *.jpg
do
    horizontal=`identify -verbose $file | grep Geometry: | awk {'print $2'} |
cut -d"x" -f 1`
    vertikal=`identify -verbose $file | grep Geometry: | awk {'print $2'} |
cut -d"x" -f 2`
    X=$Textabstandvonlinks
    Y=$(( $vertikal - $Textabstandvonunten ))
    convert -font $PfadFonts/$Schriftart -pointsize $Schriftgroesse -fill
$Schriftfarbe -draw "text $X, $Y '$Wasserzeichentext'" "$file" "`basename
Wasserzeichen_"$file`";
    echo "Bearbeite Datei $file"
done
echo "Wasserzeichen wurden erfolgreich eingearbeitet"
exit
```

Beschriftung

Mit dem Befehl

```
locate *.ttf | grep Name-der-Schrift
```

läßt sich der exakte Pfad zur gewünschten Schrift ermitteln, die im Argument *-font* angegeben werden muss. Bei einem Standard Linux Mint wäre das für die Schriftart „FreeSans“ z.B. */usr/share/fonts/truetype/freefont/FreeSans.ttf*

```
convert -font @/usr/share/fonts/truetype/freefont/FreeSans.ttf  
-pointsize 80 -fill "#ff0000" alt.png -draw "text 50, 100  
'Hier steht der Text'" neu.png
```

-pointsize gibt die Schriftgröße der unter *-font* gewählten Schriftart an. Mittels *-fill* kann eine Schriftfarbe definiert werden und *-draw* ist schließlich für die tatsächliche Beschriftung zuständig. Die beiden Werte geben den x- bzw. y-Abstand vom gerade aktiven Bezugspunkt (Änderung per *-gravity* möglich).

Text mit Schatten Variante 1

Ein Schatteneffekt wird durch das versetzte Einfügen einer Beschriftung mit unterschiedlichen Farben erzeugt. Der Befehl *-annotate* ist eine vereinfachte Variante zum Einfügen von Text mit weniger Einstellungsmöglichkeiten als *-draw*. Die Werte X + Y bewirken einen Dreheffekt.

```
convert -font @/usr/share/fonts/truetype/freefont/FreeSans.ttf  
-pointsize 50 alt.png -fill black -annotate +50+100  
'Hier steht der Text' -fill gray -annotate 0x130+50+100  
'Hier steht der Text' neu.png
```

Text mit Schatten Variante 2

Dieses Beispiel liefert eine leicht versetzte schattige Umrandung:

```
convert -font @/usr/share/fonts/truetype/freefont/FreeSans.ttf  
-pointsize 50 -fill white alt.png -stroke black -strokewidth 5  
-annotate +50+100 'Hier steht der Text'  
-stroke none -strokewidth 0 -annotate +50+100  
'Hier steht der Text' neu.png
```

Text mit einfacher Rahmenlinie

```
convert -font @/usr/share/fonts/truetype/freefont/FreeSans.ttf
        -pointsize 50 -fill white -stroke black -strokewidth 1 alt.png
        -annotate +50+100 'Hier steht der Text' neu.png
```

-*stroke* zieht eine Rahmenlinie um den Text, die Linienstärke kann mit *-strokewidth* angegeben werden.

Text mit doppelter Rahmenlinie

Dieses Beispiel erzeugt eine doppelte schwarze Rahmenlinie mit weißer Füllung.

```
convert
-font @/usr/share/fonts/truetype/freefont/FreeSans.ttf
-pointsize 50 alt.png -fill none -stroke black -strokewidth 3
-annotate +50+100 'Hier steht der Text'
-fill none -stroke white -strokewidth 1 -annotate +50+100
'Hier steht der Text' neu.png
```

Durchsichtiger Text

Dieses Beispiel generiert eine durchsichtige Schrift mit einer dünnen, schwarzen Rahmenlinie.

```
convert
-font @/usr/share/fonts/truetype/freefont/FreeSans.ttf
-pointsize 50 -fill none -stroke black -strokewidth 1 alt.png
-annotate +50+100 'Hier steht der Text'neu.png
```

Text mit „Mahjongg-Effekt“

```
convert
-font @/usr/share/fonts/truetype/freefont/FreeSans.ttf
-pointsize 50 -fill white alt.png
-stroke black -strokewidth 25 -annotate +50+100 'Hier steht der Text'
-stroke white -strokewidth 20 -annotate +50+100 'Hier steht der Text'
-stroke black -strokewidth 15 -annotate +50+100 'Hier steht der Text'
-stroke white -strokewidth 10 -annotate +50+100 'Hier steht der Text'
-stroke black -strokewidth 5 -annotate +50+100 'Hier steht der Text'
-stroke none -strokewidth 0 -annotate +50+100 'Hier steht der Text'
neu.png
```

Weitere Informations- und Ideenquellen

Die Communities

Natürlich stehen die Offenburger Communities gern bereit, wenn es um Fragen oder Anwendungsprobleme geht.
Gerne bei unserem regelmäßigen Treffen oder im Offenen Computerraum (TiP).

Internetquellen

<https://www.imagemagick.org/Usage/>

Schlussbetrachtung

Bildmanipulation auf der Kommandozeile erscheint einem „normalen“ Anwender zunächst bizarr oder umständlich.

Hat man sich jedoch einmal mit den vielfältigen Möglichkeiten und der Mächtigkeit des Paketes ImageMagick beschäftigt, zeigt sich, dass es entgegen der landläufigen Meinung sehr wohl sehr bequem und vor Allem schnell Möglichkeiten bereitstellt, die in der grafischen Welt ihresgleichen suchen.

Es sei hier noch erwähnt, dass es seit 2002 einen Fork dieser Software namens „GraphicsMagick“ gibt, welcher auch aktiv weiterentwickelt wird. Die Intention ist die Selbe, es wird jedoch behauptet, dass GraphicsMagick in einigen Punkten überlegen sei. Dies kann man auf der Webseite nachlesen und das Paket auch gerne mal ausprobieren.

<http://www.graphicsmagick.org/>

