Rhinozelfant Dokumentation

Bundeswettbewerb Informatik 2016

Johannes Sonn
2016

Inhaltsverzeichnis

Lösungsidee

Es wurde im Urwald von Informatien ein sogenannter Rhinozelfant gefunden. Dieser Rhinozelfant kann sich wie ein Chamäleon die Farbe der Haut an die Umgebung anpassen, um sich zu tarnen. Mit einer hochauflösenden Kamera, wird eine Hautschuppe des Rhinozelfants über mehrere Pixel dargestellt. Also haben benachbarte Pixel einer Schuppe des Rhinozelfants die exakt selbe Farbe und so auch dieselben RGB-Werte. Diese Pixel mit denselben RGB-Werten, sollen durch ein Programm weiß eingefärbt, also die RGB-Werte auf 255 gesetzt werden.

Um bestimmen zu können, ob ein Pixel vermutlich einem Rhinozelfant darstellt, sollen dessen RGB-Werte mit denen seines Nachbarn verglichen werden. Wenn die RGB-Werte übereinstimmen, sollen alle drei RGB-Werte, beider Pixel auf den Maximalwert 255 geändert werden.

Umsetzung

Als Hochsprache wurde Python verwendet. Das Programm wird durch die Kommandozeile aufgerufen.

Das, durch die Argumente angegebene, Input-File wird als *inputIm* geöffnet und als *inputPix* geladen. Anschließend wird eine Kopie des Input-Bildes, mit dem Namen *outputIm* angefertigt und als *outputPix* geladen.

Danach wird das Input-Bild analysiert, indem die RGB-Werte eines bestimmten Pixels mit den RGB-Werten des rechts und des oberhalb liegenden Pixels verglichen werden. Wenn die Werte identisch sind, werden die Pixel in der Kopie an derselben Position weiß gefärbt. Dies wird für alle Pixel, bis auf die Pixel in der obersten Reihe und in der Reihe am rechten Rand, durchgeführt.

Nachdem das Bild analysiert wurde, wird es unter dem angegebenen Namen und Pfad gespeichert. Das Programm muss mit den Argumenten „-i“ und „-o“ aufgerufen werden. Dabei muss dem Argument „-i“ der Pfad und Name der Input-Datei und auf „-o“ der Pfad und Name der Output- bzw. Loesungs-Datei folgen.

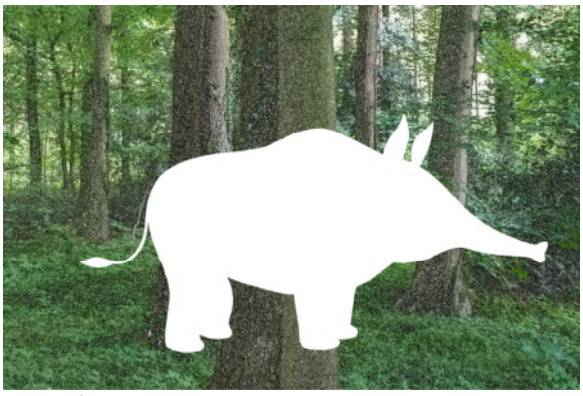
Falls das Programm mit dem Argument *–h* oder *–help* aufgerufen wird, wird ein Hilfetext ausgegeben der Beschreibt, wie das Programm richtig aufgerufen wird.

Das Programm wurde mit den Bildformaten BMP, PNG und PPM getestet. Theoretisch sollte das Programm alle Bildformate unterstützen, die von der Python Image Library (PIL) unterstützt werden.

Beispiele

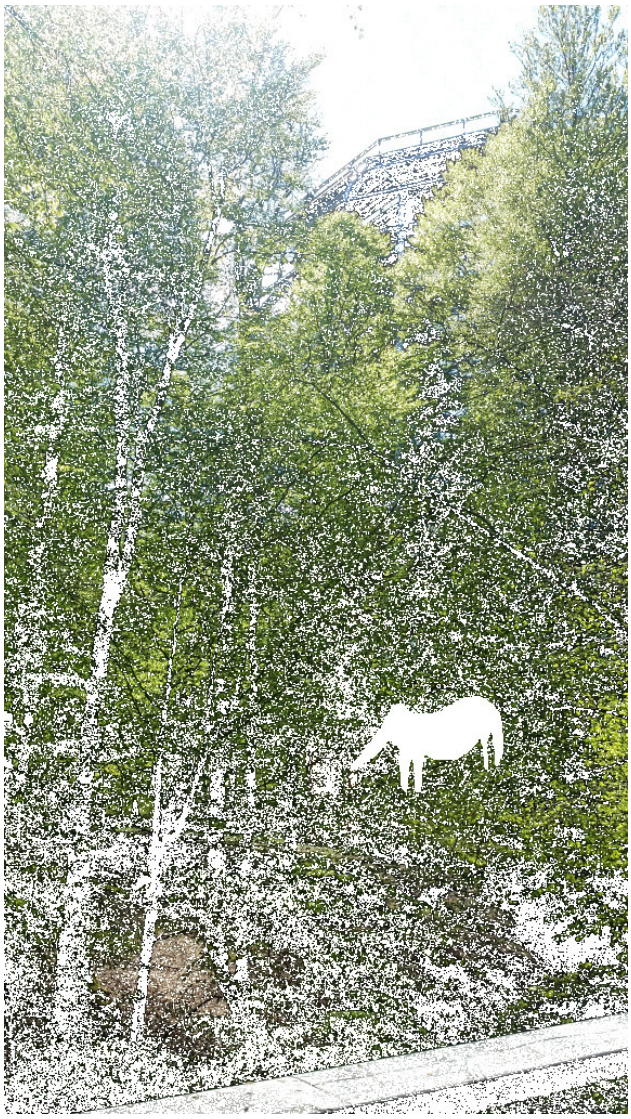
BMP-Dateien

```
$ python Rhinozelfant_015.py -i rhinozelfant1.bmp -o rhinozelfant1_Loesung.bmp
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant1_Loesung.bmp"
--------
Fertig!
$ python Rhinozelfant_015.py -i rhinozelfant2.bmp -o rhinozelfant2_Loesung.bmp
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant2_Loesung.bmp"
--------
Fertig!
$ python Rhinozelfant_015.py -i rhinozelfant3.bmp -o rhinozelfant3_Loesung.bmp
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant3_Loesung.bmp"
--------
Fertig!
```



Rhinozelfant2_Loesung.bmp

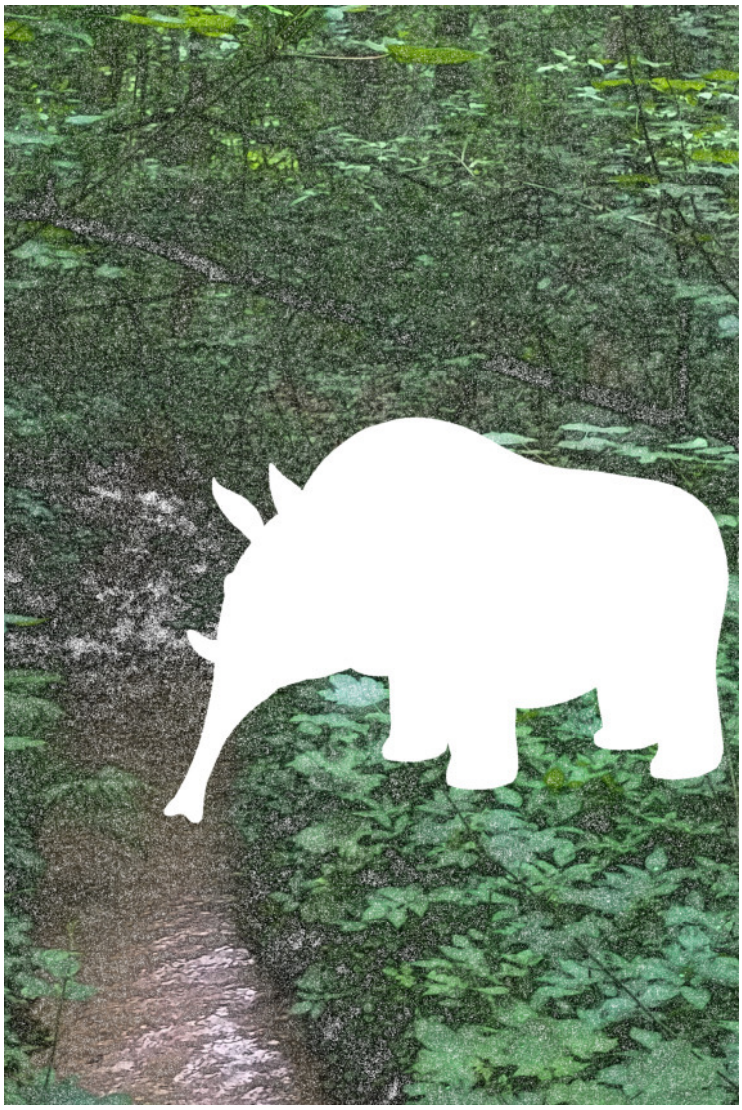
Rhinozelfant3_Loesung.bmp



Rhinozelfant1_Loesung.bmp

PNG-Dateien

```
$ python Rhinozelfant_015.py -i rhinozelfant4.png -o rhinozelfant4_Loesung.png
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant4_Loesung.png"
--------
$ python Rhinozelfant_015.py -i rhinozelfant5.png -o rhinozelfant5_Loesung.png
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant5_Loesung.png"
--------
Fertig!
$ python Rhinozelfant_015.py -i rhinozelfant6.png -o rhinozelfant6_Loesung.png
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant6_Loesung.png"
--------
Fertig!
```



Rhinozelfant4_Loesung.png

Rhinozelfant5_Loesung.png

Rhinozelfant6_Loesung.png

PPM-Dateien

```
$ python Rhinozelfant_015.py -i rhinozelfant7.ppm -o rhinozelfant7_Loesung.ppm
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant7_Loesung.ppm"
--------
Fertig!
$ python Rhinozelfant_015.py -i rhinozelfant8.ppm -o rhinozelfant8_Loesung.ppm
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant8_Loesung.ppm"
--------
Fertig!
$ python Rhinozelfant_015.py -i rhinozelfant9.ppm -o rhinozelfant9_Loesung.ppm
Importiere Photo
Analysiere Photo
Speichere bearbeitetes Photo unter "rhinozelfant9_Loesung.ppm"
--------
Fertig!
```

(Beispiel-Bilder sind in dem ZIP-Archive zu finden)

```
1   # -*- coding: utf-8 -*-
2
3   ###Titel:       Rhinozelefant
4   ###VersionsNr:  15
5   ###Autor:       Johannes Sonn
6   ###Datum:       15.01.2016
7
8   ###Import###
9   from PIL import Image
10  import time
11  import sys
12
13
14  ###Variablen deklarieren###
15  i = 0
16  falscheEingabe = True
17  usage = 'USAGE: Rhinozelefant_xxx.py -i [input-file] -o [output-file]'
18  inputIm = 0
19  outputIm = 0
20  x = 0
21  y = 0
22
23
24  ###Es wird die Anzahl der Argumente überprüft
25  if len(sys.argv) == 5:
26
27      #Falls das erste Argument gleich "-i" ist,...
28      if sys.argv[1] == "-i" or sys.argv[1] == "-I":
29
30          ###Ausgabe: "Importiere Photo"
31          print "Importiere Photo"
32
33          #...dann wird das angegeben Bild als "inputIm" geoeffnet und als "inputPix"
            geladen...
34          inputIm = Image.open(sys.argv[2])
35          inputPix = inputIm.load()
36
37          #...und es wird eine Kopie mit dem Namen "outputIm" angefertigt und als
            "outputPix" geladen.
38          outputIm = inputIm.copy()
39          outputPix = outputIm.load()
40
41
42          ###Ausgabe: "Analysiere Photo"
43          print "Analysiere Photo"
44
45          #Solange der y-Wert kleiner als die Bildgröße ist...
46          while y < inputIm.size[1]:
47              #...und der x-Wert kleiner als die Bildgröße ist...
48              while x < inputIm.size[0]:
49
50                  #...und solange er nicht am Rand ist, ...
51                  if x < inputIm.size[0] - 1:
52
53                      #...werden die RGB-Werte des Pixels[x, y] mit dem
                        Nachbar-Pixel[x+2, y] verglichen
54                      if inputPix[x, y] == inputPix[x + 1, y]:
55
56                          #Wenn die RGB-Werte identisch sind, werden in der
                            Output-Kopie die entsprechenden Pixel weiß (255, 255, 255)
                            gefärbt
57                          outputPix[x, y] = (255, 255, 255)
58                          outputPix[x + 1, y] = (255, 255, 255)
59
60                  #Wenn der Pixel nicht am Rand ist, ...
61                  if y < inputIm.size[1] - 1:
62
63
64
```

```
65
66
67                      #Wenn die RGB-Werte identisch sind, werden in der Output-Kopie
                        die entsprechenden Pixel weiß (255, 255, 255) gefärbt
68                      #...werden die RGB-Werte des Pixels[x, y] mit dem
                        Nachbar-Pixel[x+2, y] verglichen
69                      if inputPix[x, y] == inputPix[x, y + 1]:
70
71                      #Wenn die RGB-Werte identisch sind, werden in der Output-Kopie
                        die entsprechenden Pixel weiß (255, 255, 255) gefärbt
72                          outputPix[x, y] = (255, 255, 255)
73                          outputPix[x, y+1] = (255, 255, 255)
74
75                  #Nach jedem Pixel wird der x-Wert um 1 erhöht
76                  x  += 1
77
78              #Nach einer Pixel-Reihe wird der y-Wert um 1 erhöht und der x-Wert auf 0
                gesetzt
79              y += 1
80              x = 0
81
82
83          #Wenn das dritte Argument gleich "-o" ist,...
84          if sys.argv[3] == "-o" or sys.argv[3] == "-O":
85
86              #...wird ausgegeben, in welchem Pfad und unter welchem Namen die
                Output-Kopie gespeichert wird...
87              print 'Speichere bearbeitetes Photo unter "' + sys.argv[4] + '"'
88
89              #...und unter dem angegeben Pfad und Namen gespeichert
90              outputIm.save(sys.argv[4])
91
92
93              #Ausgabe 'Fertig!'
94              print "--------"
95              print 'Fertig!'
96
97
98
99      #Wenn das Programm mit dem Argument "-h" oder "--help" aufgerufen oder falsch
        aufgerufen wird,
100     #wird ausgegeben wie man es richtig aufruft
101     else:
102         print usage
103 else:
104         if sys.argv[-1] == "-h" or sys.argv[-1] == "-H" or sys.argv[-1] == "--help":
105             print usage
106         else:
107             print usage
108
109
```