Beispiellösung: Rhinozelfant

Hinweis:

Der Aufgabentext wird hier nur der Vollständigkeit halber abgedruckt. Die Dokumentation zu einer Aufgabenbearbeitung muss und soll den Aufgabentext nicht enthalten.

Im Urwald von Informatien ist ein Rhinozelfant gefunden worden. Die Forscher sind begeistert: Zum ersten Mal seit Jahrhunderten wurde ein bislang unbekanntes Großtier entdeckt. Es stellt sich die Frage, warum diese Tiere solange verborgen geblieben sind. Offenbar verfügen sie über einen besonders guten Tarnmechanismus.

Nachdem die ersten Rhinozelfanten in einen Zoo gebracht wurden, stellten die Tierpfleger fest, dass diese – ähnlich wie Chamäleons – die Farbe ihrer Haut an beliebige Umgebungen anpassen können. Wenn ein Rhinozelfant merkt, dass es beobachtet wird, nimmt jede seiner Hautschuppen die Umgebungsfarbe der dem Betrachter gegenüberliegenden Körperseite an. Dadurch wird es für den Beobachter quasi „durchsichtig".

Mit modernen Digitalkameras ist es allerdings möglich, Rhinozelfanten zu erkennen: Wenn die Auflösung nur hoch genug ist, wird jede Hautschuppe durch mehrere nebeneinander liegende Pixel dargestellt. Weil alle diese Pixel dieselbe Schuppe abbilden, haben sie genau die gleiche Farbe. Dadurch kann man feststellen, welche Pixel in einem Bild möglicherweise einen Rhinozelfanten darstellen.

Es sollen nun mehrere Fotos aus dem Rhinozelfantenwald geprüft werden, ob darauf vielleicht ein Rhinozelfant abgebildet ist.

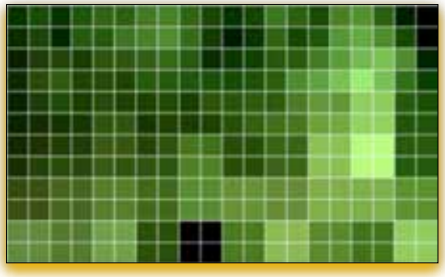
Aufgabe

Erstelle ein Programm, das diejenigen Pixel eines Bildes weiß färbt, die zu einem Rhinozelfanten gehören könnten.

Unter bwinf.de/bundeswettbewerb sind einige Bilder zur Verfügung gestellt[1]. Auf welchen ist ein Rhinozelfant abgebildet?

Lösungsidee

Für diese Aufgabe muss man wissen, dass Digitalbilder aus vielen Bildpunkten (Pixeln) bestehen, die jeweils eine bestimmte Farbe haben. Wenn man ein Digitalbild ausreichend vergrößert betrachtet, kann man die einzelnen Pixel erkennen. Hier ist ein vergrößerter Ausschnitt aus einem der Beispielbilder für diese Aufgabe (Kontrast ein wenig erhöht):

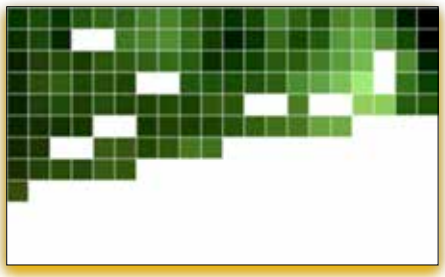


Bei genauem Hinsehen fällt auf, dass im oberen Bereich des Ausschnittes jedes Pixel eine leicht andere Farbe hat, während unten immer vier Pixel die gleiche Farbe haben. Dies sind die Rhinozelfantenschuppen.

Die Aufgabenstellung sagt nichts darüber aus, ob Rhinozelfantenschuppen auf den Bildern immer 4 Pixel groß sind. Daher können im Prinzip alle Pixel zu einem Rhinozelfanten gehören, die mindestens ein benachbartes Pixel haben, das die gleiche Farbe hat.

Eine Lösungsidee für diese Aufgabe ist also, jedes Pixel in dem Bild weiß zu färben, das ein benachbartes Pixel in der gleichen Farbe hat. Dabei werden möglicherweise auch solche benachbarte Pixel weiß gefärbt, die nicht zu einem Rhinozelfanten gehören, aber zufällig die gleiche Farbe haben. Das macht aber nichts, denn die Aufgabenstellung sagt nur, dass diejenigen Pixel weiß gefärbt werden sollen, die zu einem Rhinozelfanten gehören könnten.

Bei unserem vergrößertem Ausschnitt würde nach unserer Idee also Folgendes ausgegeben:



Umsetzung

Die Lösungsidee wird in Python implementiert. Die Python Imaging Library (PIL) stellt viele Funktionen zur Bildverarbeitung zur Verfügung. Damit funktionieren das Öffnen und Speichern des Bildes und der Zugriff auf die Pixeldaten sehr einfach. Wir importieren dazu das Modul Image der PIL.

Mithilfe zweier ineinander geschachtelter For-Schleifen werden alle Pixel einzeln betrachtet. Immer wenn ein Pixel die gleiche Farbe hat wie eines seiner Nachbarpixel, färben wir beide Pixel weiß.

Da bei dieser Aufgabe alle Pixel weiß gefärbt werden sollen, die zu einem Rhinozelfant gehören könnten, müssen wir aufpassen, dass wir nicht direkt ein Pixel im Bild weiß färben, wenn wir sehen, dass es einen gleichfarbigen Nachbarn gibt. Sonst kann es passieren, dass wir bei den anderen benachbarten Pixeln nicht mehr wissen, welche Farbe das aktuelle Pixel ursprünglich hatte. Dieses Problem wird gelöst, indem nicht die Pixel im Originalbild weiß gefärbt werden, sondern in einer Kopie des Bildes ('ausgabebild'). Dadurch können wir im Originalbild immer alle Pixel in ihrer ursprünglichen Farbe vergleichen.

Zu guter Letzt wird das Ausgabebild wieder in eine Datei gespeichert.

Beispiele

Wir rufen das Programm für zwei der Beispieldateien auf und zeigen jeweils das resultierende Bild in verkleinerter Darstellung:

```
$ ./rhino.py rhinozelfant2.png ausgabe2.png
```

Es ist ein Rhinozelfant zu erkennen.

```
$ ./rhino.py rhinozelfant6.png ausgabe6.png
```

Es ist kein Rhinozelfant zu erkennen.

Quelltext

```
#!/usr/bin/env python3

# Für Bildbearbeitungen
from PIL import Image
# Für die Übergabe von Kommandozeilenargumenten
import sys

# Prüfe ob Eingabebild und Ausgabebild angegeben wurden
if len(sys.argv) != 3:
    print("Benutzung: rhino.py <eingabebild> <ausgabebild>")
    sys.exit(0)

# Öffne das Eingabebild und lade Daten in ein Image-Objekt
# sys.argv[1] ist das erste Kommandozeilenargument
bild = Image.open(sys.argv[1])

# Mache eine Kopie des Bildes zum Bearbeiten
ausgabebild = bild.copy()

# Weiß als RGB-Wert (Rot, Grün, Blau)
weiss = (255, 255, 255)

# Betrachte alle Pixel
(breite, hoehe) = bild.size

for y in range(hoehe):
    for x in range(breite):

        # Falls zwei Pixel nebeneinander die gleiche Farbe haben,
        if x+1 < breite and (bild.getpixel((x, y))
                             == bild.getpixel((x+1, y))):
            # ... färbe beide Pixel im Ausgabebild weiß.
            ausgabebild.putpixel((x, y), weiss)
            ausgabebild.putpixel((x+1, y), weiss)

        # Falls zwei Pixel übereinander die gleiche Farbe haben,
        if y+1 < hoehe and (bild.getpixel((x, y))
                            == bild.getpixel((x, y+1))):
            # ... färbe beide Pixel im Ausgabebild weiß.
            ausgabebild.putpixel((x, y), weiss)
            ausgabebild.putpixel((x, y+1), weiss)

# Bild speichern
# sys.argv[2] ist das zweite Kommandozeilenargument
ausgabebild.save(sys.argv[2])
```

1) www.bwinf.de/bundeswettbewerb/das-bwinf-archiv/archivierte-seiten/der-35-bwinf/1-runde/material-351/