Neues Bildskalierungsverfahren im Test oder: Liebling, ich habe die Bilder geschrumpft

von Dominik Wagenführ

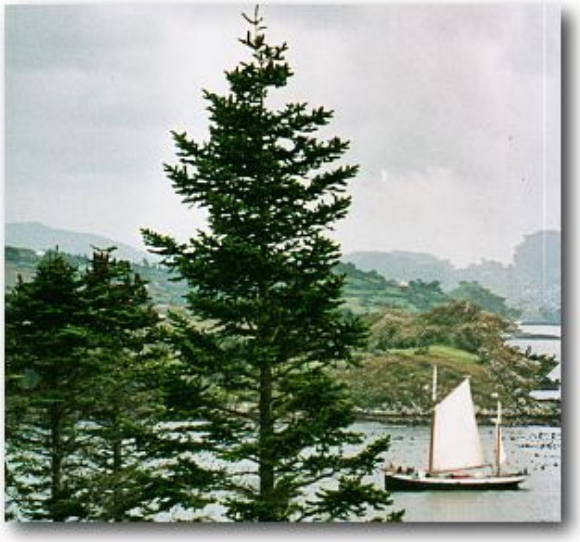
Wer kennt es nicht? Da hat man die schönen Urlaubsbilder von der Digitalkamera heruntergeladen und sieht, dass ein Großteil eines Bildes eigentlich nicht wichtig ist und man es für die Bearbeitung aber erst umständlich selbst zurechtschneiden müsste. Nun gibt es ein neues, intelligentes Bildskalierungsverfahren namens *Seam Carving* [1], das hier Abhilfe schaffen soll.

Originalbild.

Entwickelt wurde das Verfahren von Ariel Shamir und Shai Avidan, um Bilder in eine Richtung verkleinern oder auch vergrößern zu können. Dies geht natürlich auch schon jetzt in herkömmlichen Bildbearbeitungsprogrammen, nur werden hierbei die einzelnen Pixelreihen einfach nur zusammen geschoben und daraus ein neuer Bildpunkt berechnet (siehe Vergleich unten). Das führt dazu, dass alle Objekte im Bild ebenfalls ihr ursprüngliches Höhen-/Breitenverhältnis verlieren. Dagegen staucht *Seam Carving* die Bilder nicht einfach, sondern entfernt nur „ähnliche“ Bildpunkte in eine Richtung und umgeht so Objekte, die sich vom (meist gleichförmigen) Hintergrund abheben. Dies nennt sich dann „Content-Aware Image Resizing“. Zusätzlich kann man Bereiche angeben, die gesperrt sind und definitiv nicht entfernt werden sollen. Ebenso kann man Bereiche angeben, auf die beim Entfernen keine Rücksicht genommen werden muss.

Normale Skalierung auf 50 % Breite.

Je nach dem, ob man ein Bild in der Höhe oder Breite verändern will, wird vertikal oder horizontal ein Pfad („Seam“) mit minimaler Energie durch das Bild gesucht. Diese Pixel werden dann beim Verkleinern entfernt. Das bedeutet grob gesagt, es wird der Weg des geringsten Widerstandes gegangen und dieser entfernt, so dass das Bild, eine Pixelreihe oder -spalte kleiner ist. So wird fortgefahren, bis das Bild die neue Größe erreicht hat und genug Pixelreihen oder -spalten entfernt wurden. Beim Hinzufügen geht man den umgekehrten Weg und fügt die Bildpunkte des Pfades ein, den man normalerweise entfernen würde.



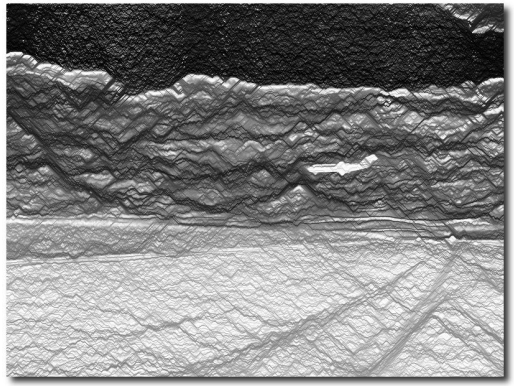
Skalierung auf 50 % Breite mit Seam Carving.

Hintergrund
Wie funktioniert das Verfahren nun genau? Die Idee ist es, Pixel mit minimaler Energie aus einem Bild zu entfernen. Neben der Energiefunktion, zu der es verschiedene Ansätze gibt, gibt es auch verschiedene Strategien, wie man Pixel entfernt:

1. Man durchsucht das ganze Bild und entfernt alle Pixel mit minimaler Energie. Das Problem ist dann, dass man pro Zeile oder Spalte nicht mehr die gleiche Anzahl von Pixel haben muss.
2. Man entfernt Pixel mit minimaler Energie, achtet aber darauf, dass man pro Zeile bzw. Spalte die gleiche Anzahl von Pixeln entfernt. Das Bild hat am Ende zwar eine Rechteckgestalt, der Inhalt ist aber verschoben und nicht mehr erkennbar.
3. Man sucht eine durchgehende Spalte oder Reihe mit minimaler Energie und entfernt diese. Dies kann zu Artefakten und Sprüngen im Bild führen, wenn wichtige Teile eines Objektes verschwinden.
4. Die Optimallösung ist es, einen zusammenhängenden Pfad mit minimaler Energie durch das Bild zu suchen, so dass pro Spalte bzw. Reihe die gleiche Anzahl von Bildpunkten entfernt werden.

Im zweiten Bild unten sind die einzelnen horizontalen Seams dargestellt, wobei Pixel mit minimaler Energie schwarz eingefärbt sind und solche mit maximaler Energie weiß. Wie man gut erkennt, wird dementsprechend bei einer vertikalen Skalierung zuerst der Himmel entfernt werden, danach die Wolken, dann der Acker und zum Schluss das Flugzeug. Das bedeutet also, dass das Flugzeug als wichtiges Objekt erkannt wurde und bei einer Skalierung erhalten bleibt. Das Bild wurde mit `mapseams` von Arachne (siehe „Implementierungen“) erstellt.

Originalbild.



Horizontale Seams (minimale Energie = schwarz).

Hat man also ein Bild I der Größe $m \times n$ gegeben, so definiert man einen vertikalen Seam als

$$s^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n,$$

so dass $|x(i) - x(i-1)| \leq 1$ für alle $i = 2, \ldots, n$, wobei $x : [1, \ldots, n] \rightarrow [1, \ldots, m]$ jeder Bildspalte eine -reihe zuordnet. Mit anderen Worten man hat in jeder Spalte genau einen Bildpunkt, so dass zwei aufeinanderfolgende Pixel keine Lücke haben. Analog dazu ist ein horizontaler Seam s^y definiert. Die Pixel auf diesem Pfad werden dann als $I_s = \{I(s_i)\}_{i=1}^n = \{I(x(i), i)\}_{i=1}^n$ bestimmt. (Die Notation wurde dem PDF [1] entnommen.)

Ist eine Energiefunktion e gegeben, so berechnet man die Gesamtkosten eines Seams als $E(s) = E(I_s) = \sum_{i=1}^n e(I(s(i))$. Gesucht ist der optimale Seam mit minimalen Energiekosten:

$$s^* = \min_s E(s) = \min_s \sum_{i=1}^n e(I(s_i)).$$

Als Energiefunktion kann man eigentlich alles nutzen, was auf die „Energie“ des Bildes eingeht. Je nach Wahl erhält man ggf. Artefakte oder der Algorithmus dauert länger. Es hat sich gezeigt, dass vor allem die Energiefunktionen e_1 und HoG („Histogram of Gradients“) gute Ergebnisse liefern.

Um den optimalen Seam zu finden, iteriert man über die einzelnen Zeilen bzw. Spalten und addiert die minimale Energie vorhergehender

und nachfolgender Punkte. Aufpassen muss man, wenn man ein Bild nicht nur in eine Richtung verkleinern will, sondern in Höhe und Breite. Die optimale Reihenfolge, ob man erst vertikale, erst horizontale oder alternierend die Seams entfernt, ist etwas komplizierter und kann im PDF [1] ab Seite 4 nachgelesen werden.

Das Vergrößern eines Bildes ist recht leicht. Man sucht hierfür der optimalen Seam, den man normalerweise entfernen würde und fügt dieses in das Bild zusätzlich ein. Dabei wird der neue Pixelwert aus der Interpolation seiner Nachbarn berechnet. Natürlich führt dieses Verfahren zu Artefakten, wenn man es einfach nur wiederholt anwenden würde, da man immerzu den gleichen Seam hinzufügt. Möchte man ein Bild um k Reihen oder Spalten vergrößern, sucht man die ersten k optimalen Seams und fügt diese neu ein. Vor allem bei extremen Vergrößerungen kann aber auch diese Methode zu Artefakten führen.

Bessere Ergebnisse erhält man, wenn man nicht die Seams des Originalbildes, sondern die Seams der Gradienten in x und y entfernt. Es kommt so zu weniger Artefakten, da auch die Übergängen zwischen zwei Bildpunkten beachtet werden.

Wer den Ausführungen jetzt nicht ganz folgen konnte, findet auf YouTube ein Video, welches das Verfahren anhand einiger Anwendungsbeispiele vorführt [2]. Das Video gibt es auch als hochauflösende Version auf Ariel Shamirs Webseite [3].

Anwendungsgebiete

Zugegeben, für die Verkleinerung privater Urlaubsbilder ist das Verfahren sicher nicht gedacht, kann aber auch hier schöne Effekte erzielen, vor allem wenn man ungewünschte Objekte beim Verkleinern entfernen will. Wozu benötigt man dies aber wirklich? Heute haben Fotografien fast immer ein Standardseitenverhältnis von 4:3. Dank Breitbildschirme und mobiler Geräte ist das 4:3-Format aber fast am Aussterben. Möchte man ein 4:3-Bild auf so einem Gerät anzeigen, hat man entweder schwarze Balken oder das Bild muss gestreckt/gestaucht werden und verliert damit das gewohnte 4:3-Verhältnis. Hierfür ist eine Objekt-erhaltende Verkleinerung sinnvoll. Zusätzlich ist es mit dem Verfahren möglich, so genannte „Multi-Size-Images“ zu erstellen. Diese enthalten die Information der einzelnen Seams direkt im Bild und können daher dynamisch und schnell ihre Größe verändern.

Eine weitere Anwendung ist das gezielte Hervorheben von Bildinhalten, ohne die Bildgröße zu verändern. So kann man ein Bild erst mit normalen Skalierungsmethoden vergrößern und dann mittels *Seam Carving* verkleinern. Die Objekte auf dem Bild wirken dann größer als vorher. Auch das Entfernen von Objekten ist so möglich. Man markiert hierzu einen Bereich, der entfernt werden soll und aus dem Bild werden dann solange Seams entfernt, bis alle markierten Pixel verschwunden sind. Danach kann man per *Seam Carving* wieder Seams einfügen, um die Originalgröße des Bildes herzustellen.

Originalbild.

Skaliertes Bild (die mittleren zwei Rehe wurden dabei entfernt).

Natürlich ist das Verfahren kein Allheilmittel und es gibt immer Bilder, bei denen die Er-

gebnisse nicht gut aussehen. Vor allem Bilder, in denen kein durchgängiger Seam gefunden wird, ohne ein wichtiges Objekt zu durchkreuzen, oder Bilder, die zuviele Bildinformationen haben, stellen ein Problem dar.

Implementierungen
Inzwischen gibt es auch einige Implementierungen des Verfahrens. Arachne [4] ist eine GTK-Anwendung, welche gleichzeitig die Bibliothek Seamstress mitbringt, die man aber auch einzeln installieren kann. Wer das Programm nutzen möchte. muss nur die Pakete **build-essential**, **libgtk2.6-dev**, **libtiff4-dev** und **libgdk-pixbuf-dev** installieren. Danach entpackt man den Quellcode und kompiliert das Programms mittels

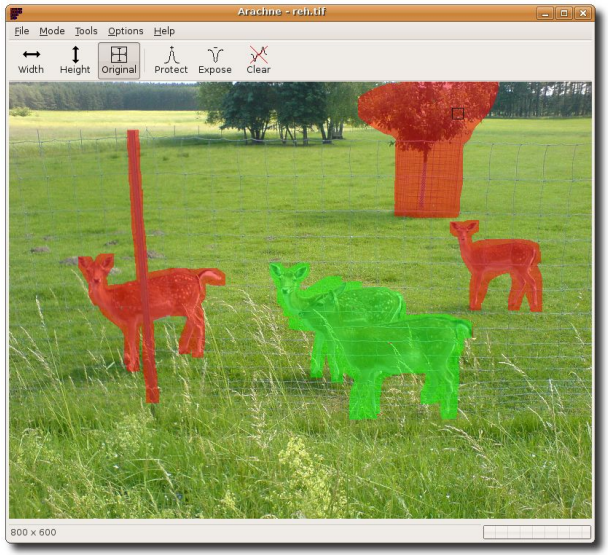
```
./configure
make
```

Danach kann Arachne per Eingabe von

```
./arachne
```

gestartet werden.

Die Benutzung ist recht intuitiv. Über die Schaltflächen „*Protect*“ bzw. „*Expose*“ bestimmt man die Regionen, die man schützen bzw. freigeben will (im Screenshot rot bzw. grün). Klick man dann auf „*Width*“ oder „*Height*“, werden zuerst alle Seams im Bild berechnet. Danach kann man das Bild in die gewählte Richtung verkleinern und sieht „live“, wie sich das Bild verändert. Das Ergebnis kann über „*File* » *SaveAs*“ auch abgespeichert werden. Aktuell ist es leider nicht möglich das Bild in beide Richtungen gleichzeitig zu verkleinern. Eine Vergrößerung wird ebenfalls nicht angeboten.



GTK-Anwendung Arachne.

Arachne bringt auch noch drei Terminalprogramme mit, mit dem man die Seams über die Kommandozeile anzeigen (`mapseams` – siehe oben), eine bestimmte Anzahl Seams entfernen (`removeseams`) oder das Bild skalieren (`retarget`) kann.

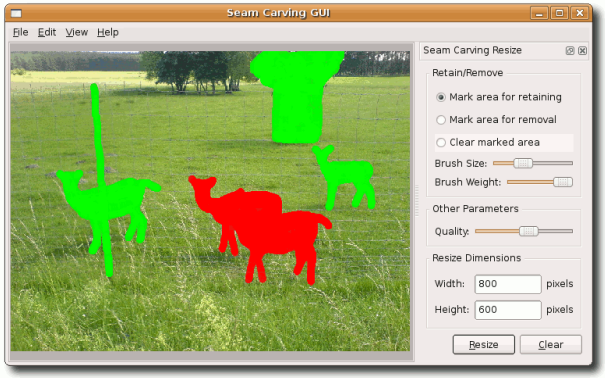
Eine zweite Implementierung gibt es von Gabe Rudy [5] namens Seam Carving GUI. Um das QT-Programm kompilieren zu können, müssen die Pakete **build-essential** und **libqt4-dev** installiert werden. Danach kann man mittels

```
qmake seam-carving-gui.pro
make
```

die Kompilierung starten und das Programm dann per

```
./SeamCarvingGui
```

starten.



QT-Anwendung Seam Carving GUI.

Die Bedienung ist wieder intuitiv. Man öffnet ein Bild per „*File* » *Open...*“, klickt auf „*Mark area for retaining*“ bzw. „*Mark area for removal*“, um die Bildbereiche grün bzw. rot zu markieren, die man behalten bzw. freigeben will, stellt danach die neue Breite („*Width*“) und Höhe („*Height*“) ein und klickt auf „*Resize*“.
Der Nachteil des Programms ist, dass die Maske (grün und rot im Screenshot) nach einer

Veränderung nicht behalten wird. Sprich, skaliert man das Bild und es entspricht nicht den Erwartungen, kann man dies zwar rückgängig machen, aber man muss die komplette Maske neu zeichnen.

Auch für das Bildbearbeitungsprogramm GIMP gibt es ein Plugin namens Liquid Rescale [6]. Nach der Installation des deb-Paketes oder Kompilierung des Quellcodes, wofür man die Pakete **build-essential**, **gettext** und **libgimp2.0-dev** benötigt, findet man das Plugin im Menüpunkt „*Ebenen » Liquid rescale...*“. Wer ein Bild einfach nur verkleinern möchte, braucht keinerlei Änderungen (bis auf die neue Bildgröße natürlich) vorzunehmen und kann sofort auf „*OK*“ klicken.

GIMP-Plugin Liquid Rescale.

Etwas komplizierter wird es, wenn man die Sonderfunktionen nutzen möchte, um Bildbereiche zu sperren oder explizit freizugeben. Hierzu öffnet man das Bild ganz normal, erzeugt eine neue Ebene über „*Ebene » Neue Ebene...*“ und malt auf dieser den Bereich aus, der gesperrt/freigegeben werden soll, die gewählte Zeichenfarbe oder -methode ist egal. Danach wählt man den Hintergrund im Ebenen-Dialog aus. klickt wieder auf „*Ebene » Liquid rescale...*“, aktiviert diesmal aber „*Elemente erhalten*“ bzw. „*Elemente verwerfen*“ und wählt die eben erstellte Ebene aus, um Bildbereiche für Veränderungen zu sperren oder explizit freizugeben.
Man kann auch beide Methoden gleichzeitig nutzen, in dem man vorher zwei neue Ebenen erstellt und die eine zum Sperren von Bereichen, die anderen zum Freigeben nutzt. Beim GIMP-Plugin funktioniert die Vergrößerung von Bildern aber nicht korrekt, so dass man keine Bildbereiche sperren kann, auch wenn man dies einstellt. Ebenso scheint es in Version 0.3.0 einen Fehler zu geben, so dass nicht alle Elemente korrekte entfernt werden.

Es gibt noch viele weitere Implementierung, unter anderem auch Online-Anwendungen, die das Verfahren umsetzen [7] [8]. Einige Umsetzungen sind aber nicht ganz optimal programmiert, so dass die Geschwindigkeit zu wünschen übrig lässt, einige Funktionen fehlen oder das Ergebnis keine gute Qualität hat.

Links

[1] http://www.faculty.idc.ac.il/arik/imret.pdf
[2] http://www.youtube.com/watch?v=6NcIJXTlugc
[3] http://www.faculty.idc.ac.il/arik/IMRet-All.mov
[4] http://seam-carver.sourceforge.net
[5] http://code.google.com/p/seam-carving-gui
[6] http://registry.gimp.org/plugin?id=10292
[7] http://www.hackszine.com/blog/archive/2007/09/open_source_seam_carving.html
[8] http://en.wikipedia.org/wiki/Seam_carving