ImageJ Plugins

Markus Fischböck (780486)

BHT Berlin - April 2012

Zusammenfassung

Beschrieben ist eine Java Anwendung zur Manipulation von Bildern unter zuhilfenahme der ImageJ Library, sowie des AWT Frameworks zur Erstellung einer grafischen Oberfläche.

Inhaltsverzeichnis

1 Einleitung

Zu erstellen war eine Anwendung, welche unter der Zuhilfenahme der ImageJ Library die Manipulation von Bildern erlaubt. Um die Anwendung komfortabel bedienen zu können wurde eine grafische Oberfläche mit WindowBuilder erstellt. Diese erlaubt es Bilder zu öffnen und manipulierte Bilder abzuspeichern. Eingabeparameter für Plugins sind über dedizierte Dialoge einstellbar. Das vorliegende Dokument enthält Informationen zu allen Plugins und stellt in kurzem die Funktionsweise einzelner Plugins vor.

2 Plugins

2.1 Channel Remover Plugins

Das Plugin zum Kanal entfernen bietet dem Benutzer die Möglichkeit einzelne Farbkanäle aus dem Eingabebild zu entfernen. Hierzu werden 3 Farbräume zur Verfügung gestellt, aus denen der Anwender auswählen kann: RGB, CMY und YUV. Die Funktionsweise der Plugins ist dabei relativ trivial. Es wird über alle Pixel des Eingabebildes iteriert und die Farbinformationen der einzelnen Kanäle extrahiert. Dabei werden die Farbwerte der zu entfernenden Kanäle auf 0 gesetzt. Da das Eingabebild zunächst nur RGB Farbinformationen enthält, muss beim entfernen von CMY oder YUV Kanälen zunächst intern in den entsprechenden Farbraum konvertiert werden. Hierzu wurden die aus Übung 1 erstellten Konverter verwendet, welche erlauben zwischen den einzelnen Farbräumen hin und her zu konvertieren.

2.2 Brightness Plugin

Das Plugin zum verändern der Helligkeit des Eingabebildes bietet dem Anwender die Möglichkeit die Helligkeit in unterschiedlichen Farbräumen zu verändern. Hierzu wurden ebenso wie beim entfernen von Kanälen die aus Übung 1 erstellten Konverter verwendet. Der Benutzer kann hier wählen zwischen RGB, HSV und YUV. Um eine komfortable Bearbeitungsmöglichkeit zu bieten, war es notwendig das Ursprungsbild nicht schon beim verschieben des Helligkeitsreglers destruktiv zu bearbeiten, sondern dem Anwender zunächst eine Vorschau zu bieten. Erst bei der Bestätigung des Dialoges sollte das Ursprungsbild entsprechend transformiert werden. Dies geschieht in der Anwendung unter Zuhilfenahme einer Kopie des Eingabebildes, welches dem Benutzer als Vorschau dient.

Die Änderungen an der Helligkeit finden in den einzelnen Farbräumen auf unterschiedliche Art und Weise statt. Unter Benutzung des RGB Farbraumes werden die Farbwerte alle Kanäle gleichermaßen in- oder dekrementiert. YUV und HSV bieten für die Helligkeitsregelung dedizierte Kanäle, welche das ändern der Helligkeit basierend auf einem einzelnen Wert erlauben. Beim YUV Farbraum ist dazu der Y-Parameter (Luminanz) zuständig. Beim HSV Farbraum reicht es aus, den V-Parameter entsprechend zu verändern.

Auffällig ist, dass beim RGB Farbraum die besten Ergebnisse erzielt werden. Eine Änderung der Helligkeit im HSV oder YUV Farbraum unter Verwendung des gleichen Eingabeparameters für die Helligkeit führt schnell zum Verlust von Farbinformationen. Dies liegt möglicherweise unter anderem an Rundungsfehlern die bei der Konvertierung in die entsprechenden Farbräume auftreten.

2.3 Skalierung

Das Plugin zur Skalierung bietet dem Anwender zunächst die Möglichkeit die Ausgabegröße des Bildes anhand zweier sogenannter “Spinner” widgets einzustellen. Ein zusätzlicher Schalter erlaubt es ausserdem, die Regler so miteinander zu verbinden, dass die Proportionen des Eingabebildes erhalten bleiben. Weiter wird dem Anwender angeboten den Algorithmus, welcher zur Skalierung verwendet wird zu bestimmen. Hier werden 2 Verfahren angeboten, welche im folgenden kurz erklärt werden.

2.3.1 Nearest Neighbor

Bei der Skalierung unter Verwendung des Nearest Neighbor Verfahrens, wird zunächst die Position des Ausgabepixels ermittelt. Dies geschieht unter Verwendung folgender Gleichungen:

$$x = \frac{x'(w-1)}{(w'-1)} \qquad y = \frac{y'(h-1)}{(h'-1)} \tag{1}$$

Die Farbinformation des hiermit berechneten Punktes wird nun durch Runden der Werte und Extraktion aus dem Quellbild ermittelt. Das Verfahren selbst ist sehr schnell, beinhaltet aber den Nachteil, dass bei Skalierung mit hohen Größenunterschieden Artefakte auftreten.

2.3.2 Bilinear

Bei der bilineraren Interpolation wird die Position des Augabepixels zunächst so berechnet, wie das auch beim Nearest Neighbor Verfahren der Fall ist. Hierbei wird der Farbwert des neuen Pixels allerdings nicht durch Runden der resultierenden Werte ermittelt, sondern es wird zwischen 4 Punkten interpoliert. Da das Augabepixels irgendwo zwischen 4 Eingabepixeln liegt, muss der Farbwert anteilig berechent werden. Dabei werden zunächst in horizontaler Richtung die Farbwerte ermittelt und deren Resultate nochmals zwischen den beiden vertikalen Pixeln anteilig berechnet. Das Verhältnis der Farbwerte zueinander wird dabei über folgende Gleichung ermittelt:

$$n_\mathrm{x} = x - \lfloor x \rfloor \qquad n_\mathrm{y} = y - \lfloor y \rfloor \tag{2}$$

Ausgehend von diesen Verhältnissen, werden die horizontalen Farbwerte ermittelt:

$$\begin{aligned} a &= (1 - n_\mathrm{x}) \cdot I(\lfloor x \rfloor, \lfloor y \rfloor) + n_\mathrm{x} \cdot I(\lceil x \rceil, \lceil y \rceil) \\ b &= (1 - n_\mathrm{x}) \cdot I(\lfloor x \rfloor, \lceil y \rceil) + n_\mathrm{x} \cdot I(\lceil x \rceil, \lceil y \rceil) \end{aligned} \tag{3}$$

Anschließend fließen die resultierenden Werte in die vertikale Interpolation mit ein:

$$p = (1 - n_\mathrm{y}) \cdot a + n_\mathrm{y} \cdot b \tag{4}$$

2.4 Histogramm

Das Histogram Plugin ermittelt die Helligkeitswerte des Eingabebildes und gibt das resultierende Histogramm des Bildes in einem Dialog wieder. Zur Erstellung des Graphen wurde auf die Library JFreeChart zurückgegriffen, welche für das erstellen von Histogrammen eigene Methoden anbietet. Der Benutzer hat bei der Erstellung des Histogramms die Möglichkeit dies, für unterschiedliche auswählbare Farbräume zu tun. Angeboten werden hierfür RGB, YUV und HSV. Zwar wird die Bibliothek JFreeChart unter der LGPL angeboten, jedoch ist eine ausführliche Dokumentation leider nur gegen Bezahlung erhältlich, was zu Einschränkungen in der Erstellung des Diagramms führte. Geplant war hierfür eigentlich alle Kanäle separat anzuzeigen. Per Default wird im Hintergrund auch ein Schatten der Kurve gezeichnet, was eher störend wirkt, dies konnte jedoch aufgrund mangelnder Dokumentation ebenfalls nicht angepasst werden.

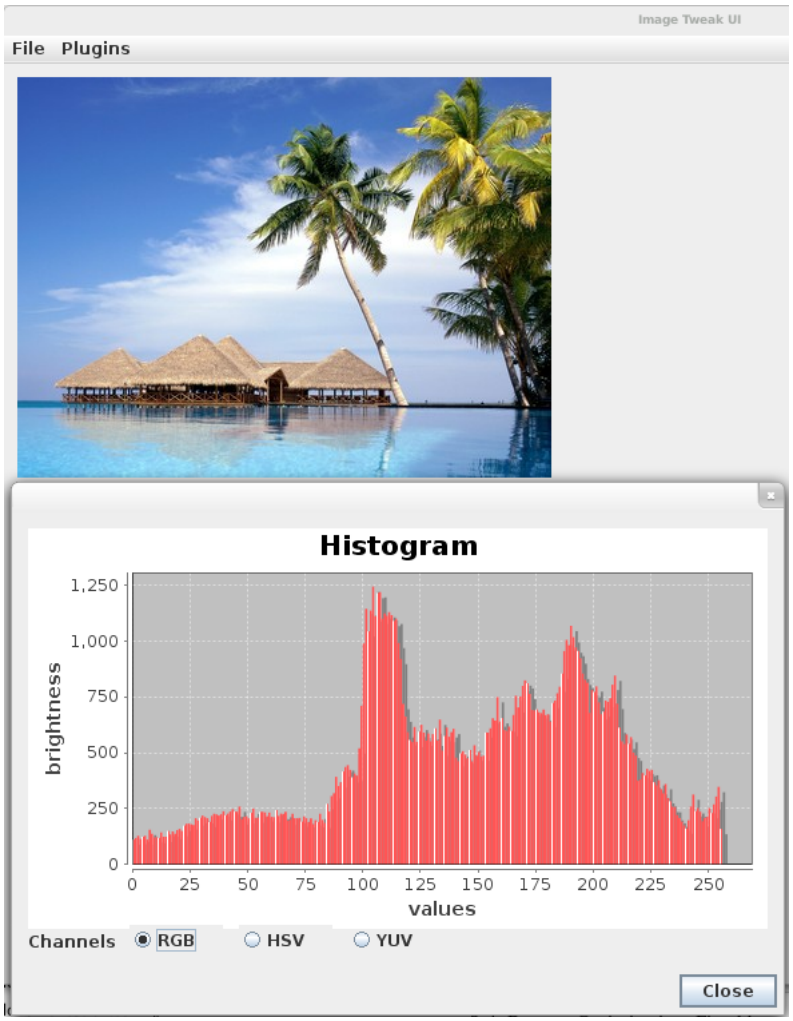

Abbildung 1: ImageTweak UI - RGB Histogram

2.5 Blur Filter

Der Blur Filters wurde mit Hilfe einer Konvolutionsmatrix implementiert. Hierbei wird über jedes Pixel im Bild iteriert und der neue Farbwert als Mittel des aktuellen Pixels und aller seiner Nachbarn berechnet. Entscheidend ist dabei, nicht über die Grenzen des vorhandenen Bildes zuzugreifen. Ein erster Ansatz mit 2 Schleifen, welche in X und Y Richtung iterieren war von Fehlern geprägt. Nach einer Überarbeitung des Algorithmus und der Begrenzung auf eine Richtung - was vollkommen ausreichend ist - konnten diese Fehler beseitigt werden.

3 Erweiterungen

Aufgrund von Zeitmangel war es mir nicht möglich zusätzliche Aufgabenstellungen zu bearbeiten. Jedoch war bei der Planung ein zentraler Punkt, eine Anwendung mit einer grafischen Oberfläche zu erstellen. Zwar bot ImageJ hierfür rudimentäre Möglichkeiten um einfache Dialoge zu erstellen, der Fokus lag aber in meiner Implementierung darauf alles in einer komfortabel zu bedienenden GUI darzustellen. Aufgrund dieser Entscheidung zu Beginn der Aufgabe, fiel das Augenmerk schnell auf das Eclipse Plugin “Window Builder” mit dem per WYSIWYG Editor auf schnelle und unkomplizierte Weise grafische Oberflächen erzeugt werden können. Der Mehraufwand der Entwicklung eines GUI stelle sich allerdings im Laufe der Bearbeitung der Aufgabe als sehr zeitraubend heraus, woraus ich für kommende Aufgaben schließe, dass es vernünftiger ist näher an der eigentlichen Aufgabenstellung zu arbeiten.