
Programmierpraktikum

Einführung in die Bildverarbeitung

Auf diesem Blatt geht es darum, vier neue Operatoren in das Framework zu integrieren:

ColorConverter Konvertierung zwischen verschiedenen Bildformaten

RGBSplitter Aufsplitten der verschiedenen Kanäle eines RGB-Bildes

RGBMerger Zusammenfügen mehrerer einkanaliger Bilder zu einem RGB-Bild

HSVManipulator Operator zum Testen von HSV-Operationen

Aufgaben

1. Erstellen Sie die Klasse **ColorConverter**. Einige Operatoren setzen ein bestimmtes Bildformat (z.B. Grauwertbild) beim Eingabebild voraus. Daher ist es nötig, einen Operator zu implementieren, der die Umrechnung von einem Format in alle anderen Formate ermöglicht. Diese Aufgabe übernimmt der Operator **Color Converter**.

Wie bekannt gibt es die folgenden Bildtypen:

- BINARY
- GRAY
- RGB
- HSV

Durch den **ColorConverter** sollen beliebige Umwandlungen zwischen den bekannten Datentypen möglich sein. Beachten Sie folgendes:

- Die Umrechnung von RGB nach HSV und umgekehrt wird recht gut unter <http://wikipedia.de> beschrieben. Sie können die Umrechnung überprüfen, indem Sie die berechneten Werte mit denen eines Bildbearbeitungsprogrammes, wie z.B. Gimp vergleichen.
- Die Umrechnung zu einem Binärbild kann für alle Formate erfolgen, indem der neue Pixelwert immer auf 1 gesetzt wird, ausser das Eingabepixel war schwarz (in diesem Fall ist das neue Pixel 0).
- Bei der Umrechnung von RGB nach Grauwertbild werden die einzelnen Kanäle verschieden stark gewichtet. Recherchieren Sie dazu in der Literatur oder dem Internet.
- Der Eingabetyp ergibt sich direkt aus dem Bild auf dem Eingabeport. Der gewünschte Ausgabebetyp soll über den Parameter **outputType** gesteuert werden, der sinnvollerweise einen **enum**-Datentyp besitzt. Zur Bearbeitung von Parametern dieses Typs wird ein **EnumEditor** benötigt, s. dazu untenstehende Aufgabe.

Erweitern Sie ausserdem die zur Darstellung eines Bildes benötigten Klassen so, dass alle Bildtypen im `PortViewerPanel` angezeigt werden können.

2. Erstellen Sie den Operator `RGBSplitter`. Dieser soll nun aus einem RGB-Bild die einzelnen Kanäle (jeweils als Grauwert-Bild) zurückgeben. Der Operator hat somit einen Eingabeport, drei Ausgabeports und keine Parameter.
3. Erstellen Sie den Operator `RGBMerger`. Der Operator `RGBMerger` ist das Gegenstück zum Operator `RGBSplitter`. Er erhält als Eingabe drei Grauwertbilder und erzeugt daraus ein RGB-Bild. Testen Sie die beiden Klassen, indem Sie z.B. die RGB-Kanäle eines Bildes vertauschen.
4. Erstellen Sie die Klasse `HSVManipulator`, mit dessen Hilfe Sie sich mit dem HSV-Farbraum näher vertraut machen. Implementieren Sie verschiedene Methoden, die die Kanäle des HSV-Bildes verändern, um z.B. die Helligkeit des Bildes zu erniedrigen, oder bestimmte Farbbereiche auszublenden. Fügen Sie selbständig sinnvolle Parameter hinzu. In dieser Aufgabe sollen Sie zeigen, dass Sie mit dem HSV-Farbraum vertraut sind und die Bedeutung der einzelnen Kanäle kennen - lassen Sie sich was einfallen!
5. Erstellen Sie die Klasse `EnumEditor` im Package `jimpro.parametereditor`. Das Framework unterstützt bisher nur die Eingabe von `int`-Parametern und von Dateinamen. Bei vielen Operatoren (wie z.B. den Color Converter) hat man die Auswahl zwischen verschiedenen, vorgegebenen Modi. Es bietet sich an, die Modi als `enum` darzustellen¹.

Beispiel:

Ein Operator benötigt z.B. einen Parameter, der die Werte `BORDER`, `INSIDE` und `OUTSIDE` annehmen kann. Dazu wird ein `enum`-Typ definiert:

```
public enum RegionPolicy {
    BORDER, INSIDE, OUTSIDE
}
```

Zusätzliche erhält der Operator einen Parameter `policy`

```
// inside the operator...
@Parameter RegionPolicy policy = RegionPolicy.BORDER; // default value
```

Mit dem `EnumEditor` soll es nun möglich sein, die Werte von `policy` zu bearbeiten.

Als Eingabewidget ist z.B. eine `ComboBox` (`JComboBox`) sinnvoll, in der der gewünschte Wert selektiert wird. Die Klasse `EnumEditor` dient zum Editieren beliebiger `enum`-Typen. Beachten Sie folgendes:

- Alle Aufzählungstypen sind in Java automatisch von der Klasse `java.lang.Enum` abgeleitet.
- Den Typ (also eine Instanz von `java.lang.Class`) des zu bearbeitenden Parameters erhalten Sie durch den Aufruf der Methode `getType()` auf einem Objekt vom Typ `ParameterSpec` - in diesem Fall wird das der Typ einer von `java.lang.Enum` abgeleiteten Klasse sein (z.B. `RegionPolicy.class`)

¹Aufzählungstypen, die nur diskrete vorbestimmte Werte annehmen können

- Jeder Parametereditor erhält über `getParameterValue()` automatisch den aktuellen Wert des Operatorparameters und kann ihn durch Aufruf von `setParameterValue()` setzen.
- Jede `enum`-Klasse besitzt automatisch die statische Methode `values()`, die alle Werte des Parameters als Array zurückgibt, z.B. so:

```
RegionPolicy[] values = RegionPolicy.values();
```

Da der `EnumEditor` jedoch mit beliebigen `enum`-Typen arbeiten soll, funktioniert dieser Ansatz hier nicht sofort (da man die statische Methode `values` nicht aufrufen kann). Lösen Sie dieses Problem durch Verwendung der Java-Reflection-API².

- Damit alles funktioniert, muss ein entsprechender Abschnitt für `EnumEditor` in die Methode `getParameterSpec()` der Klasse `Operator` eingefügt werden.

Viel Spaß!

²Das Internet hilft Ihnen weiter, falls Sie damit nicht vertraut sind ;-)