3. Rauschen

Rauschen

In den folgenden Übungen werden wir verschiedene Filter programmieren. Um die Wirkung dieser Filter testen zu können, benötigen wir nun eine Möglichkeit, Bilder gezielt zu verrauschen. In der Bildverarbeitung unterscheidet man im Wesentlichen zwei Arten von Rauschen:

Gauss-Rauschen Verursacht durch die Bildaufnahme und Bildübertragung ist prinzipiell jedes Bild mit einem mehr oder weniger starken Gauss-Rauschen überlagert. Oft ists dieses Rauschen mit dem Auge nicht erkennbar, aber die zahlenmäßige Ausgabe der Pixelwerte macht dies dennoch deutlich.

Salz- und Pfeffer-Rauschen Es kann vorkommen, dass einzelne Pixel im Kamera-Chip defekt sind und unabhängig von der Szene ein konstantes Signal liefern. Je nachdem, ob die Pixel konstant schwarz oder weiß oder beides sind, wirkt dies, wie aufgestreutes Salz und oder Pfeffer auf einem Foto.

Implementieren Sie wahlweise eine der beiden Rausch-Arten nur für Grauwertbilder.

Gauss-Rauschen

Beachten Sie, dass man unter Gausch-Rauschen eine Normalverteilung versteht, die durch Mittelwert und Varianz gekennzeichnet ist. Falls der Mittelwert des Rauschens gleich null ist, bleibt der Mittelwert des Bildes unverändert. Die Varianz ist ein Maß für die Stärke des Rauschens.

Gauss-Rauschen ist additiv, d.h. die positiven oder negativen Werte des Rauschgenerators werden zum originalen Pixelwert hinzu addiert. Dabei ist zu beachten, dass so der Wertebereich [0,255] über- oder unterschritten werden kann.

Die C++-Funktion `rand()` liefert keine Normalverteilung, sondern eine Gleichverteilung, d.h. alle erzeugten Zufallszahlen sind gleichwahrscheinlich. Zur Erzeugung einer Normalverteilung kann man z.B. die Standard-Funktion `std::normal_distribution` nutzen oder eine Berechnung von Normalverteilungen aus Gleichverteilungen implementieren.

Implementieren Sie eine Funktion mit folgendem Interface:

```
1 /** Adds gaussian noise to gray or RGB image
2 * \param src: source image
3 * \param sig: variance of noise, 68% of noise is inside mue +/- sig
4 * \param mue: mean of noise, unequal zero changes mean of image
5 * \return image with noise */
6 Image* NoiseGaussian(Image* src, double sig=5.0, double mue=0.0);
```

Algorithmus

1. Iterieren Sie über alle Pixel mit einer `for`-Schleife
2. Erzeugen Sie für jeden Pixel eine normalverteilte Zufallszahl
3. Addieren Sie diese Zufallszahl zum Pixelwert des Originalbildes und ...
4. ... achten Sie darauf, dass es nicht zum Überlauf des Wertebereiches kommt

Siehe:
http://www.cplusplus.com/reference/random/normal_distribution/
http://c-faq.com/lib/gaussian.html

Salz- und Pfeffer-Rauschen

Implementieren Sie eine Funktion mit folgenden Interface:

```
1 /** Adds salt- and/or pepper-noise to images, which means, that some
2  * randomly selectes pixel are set to black or white
3  * \param src: source image
4  * \param value: ratio of noisy to clean pixels in percent [0..100]
5  * \param mode: 0=Salt, 1=Pepper, 2=Both
6  * \return image with noise */
7 Image* NoiseSaltAndPepper(Image* src, int value=10, int mode=2);
```

Algorithmus

1. Iterieren Sie über alle Pixel mit einer `for`-Schleife
2. Erzeugen Sie für jeden Pixel eine gleichverteilte Zufallszahl
3. Abhängig von der Zufallszahl und dem Parameter `value` ...
4. ... übernehmen Sie den Pixelwert des Originalbildes oder ...
5. ... setzen den neuen Pixelwert auf 0 oder 255

Optional

- Implementieren Sie auch die zweite Rausch-Art.
- Implementieren Sie das Rauschen auch für RGB-Bilder