1 Seperierbare Filter

1.1 2D Faltung

Wir können den 2D-Kernel Algorithmus folgend grob in Code schreiben:

Bild der Grösse $M \times N$ und Filtermaske der Grösse $k \times k$

```
for (int i = 0; i < M; i++) {
   for (int j = 0; j < N; j++) {
         // ueber jedes Pixel im Bild iterieren -> N * M Iterierungen
      for (int p = 0; p < k; p++) {
         for (int q = 0; q < k; q++) {

            // durch jeden Eintrag des Kernels iterieren -> k *k Iterierungen

            applyFilter(i, j, p, q);
         }
      }
   }
}
```

1.2 1D Faltung

Den 1-D Kernel kann folgend grob beschrieben werden, mit einem Bild derselben Grösse und einer Filtermaske der Grösse k.

```
// zuerst die erste Faltung ausfuehren
for (int i = 0; i < M; i++) {
   for (int j = 0; j < N; j++) {
   // durch jedes Pixel des Bildes iterieren -> M * N Iterierungen
      for (int p = 0; p < k; p++) {
         // 1D Filter anwenden -> k Iterierungen
         applyFilter(i, j, p);
      }
   }
}

// dann die zweite Faltung auf das entstandene Bild
for (int i = 0; i < M; i++) {
   for (int j = 0; j < N; j++) {
      // durch jedes Pixel des Bildes iterieren
      for (int p = 0; p < k; p++) {
         // 1D Filter anwenden -> k Operationen
         applyFilter(i, j, p);
      }
   }
}
```

Empirisch lässt sich die Laufzeit zeigen, indem wir am Beispiel der Unit-Tests die Iterationen zählen. Da es sich bei beiden Tests um dasselbe Bild handelt, ist bei beiden der Eingabewert $n = M \cdot N = 480 \cdot 500 = 240000$.

Bei der 1D-Faltung wird einmal horizontal und einmal vertikal gefiltert, wobei beide Male über das gesamte Bild, also n-Mal iteriert wird, wobei pro Iteration $k = 5$ mal über den Kernel iteriert wird. Dies entspricht gesamthaft $2 \cdot n \cdot k = 2 \cdot 240000 \cdot 5 = 2400000$ Iterationen. Daraus lässt sich eine Laufzeit von $\Theta(2nk)$ ableiten, was $\mathcal{O}(n)$, also einer linearen Laufzeit entspricht, da bei $\mathcal{O}$ Konstanten wie $2k$ ignoriert werden.

Bei der 2D-Faltung wird nur einmal über das gesamte Bild iteriert, dabei wird pro Iteration einmal über den Kernel iteriert, also $k \cdot k = 25$ Mal. Somit wird gesamthaft $nk^2 = 480000 \cdot 25 = 6000000$ mal iteriert. Daraus lässt sich die Laufzeit $\Theta(nk^2)$ ableiten, was auch einer linearen Laufzeit von $\mathcal{O}(n)$ entspricht. Da die Konstante k^2 für Werte $k > 2$ grösser ist als die Konstante $2k$ handelt es sich bei der 1D-Faltung um eine konstant optimierte Laufzeit.

2 Filtern im Fourierraum

Eine Faltung im Ortsraum entspricht einer Multiplikation im Fourierraum, wie auch andersrum. Dies bedeutet, dass wir die Anwendung der Kernel anstatt durch eine Faltung im Ortsraum (wie in den Übungen implementiert) auch durch eine Multiplikation im Fourierraum implementieren könnten. Dafür müssten wir zuerst das Bild und den Kernel in den Fourierraum transformieren, dort die beiden Funktionen miteinander multiplizieren und die daraus resultierende Funktion zurück in den Ortsraum transportieren. Die absolute Laufzeit des 2DFFT beträgt $(MN)^2 log_2(MN)$, somit beträge die absolute Laufzeit dieses Prozesses $2(MN)^2 log_2(MN) + MN$. Die $\mathcal{O}$-Laufzeit bleibt gleich, da Konstanten ignoriert werden.

Es muss also gelten, dass die Laufzeit der Multiplikation im Fourierraum geringer ist, falls $2(MN)^2 log_2(MN) + MN > MNk^2 \Leftrightarrow 2MN + 1 > k^2$ gilt. Wenn also $2MN + 1 > k^2$ gilt, dann lohnt sich der Umweg über den Fourierraum aus einer Laufzeitorientierten Perspektive, obwohl die $\mathcal{O}$-Notation dies nicht veranschaulicht.