

SBV1, Signal- und Bildverarbeitung – WS 2022

Übungsabgabe 1b

Rudolf Hofmeister / Claudia Klausgraber / David Lang

20. November 2022

Zusammenfassung

Rasterentfernung im Frequenzraum, Anisotrope Diffusion, Stressanalyse, RL-Deconvolution

1.5 Rasterentfernung im Frequenzraum

a) Lösungsidee

Zur manuellen Manipulation von Bildern im Frequenzraum wurden 3 selbst gewählte Test-Bilder mit regulären Rastern untersucht. Die Manipulation wurde in ImageJ mithilfe von verschiedenen Formen durchgeführt. Ziel dabei war es, die Stellen im Frequenzraum zu eliminieren, welche das Raster verursachen. Dabei wurde auf Abweichungen geachtet, welche in einem natürlichen Bild ohne regulärem Raster nicht vorkommen.

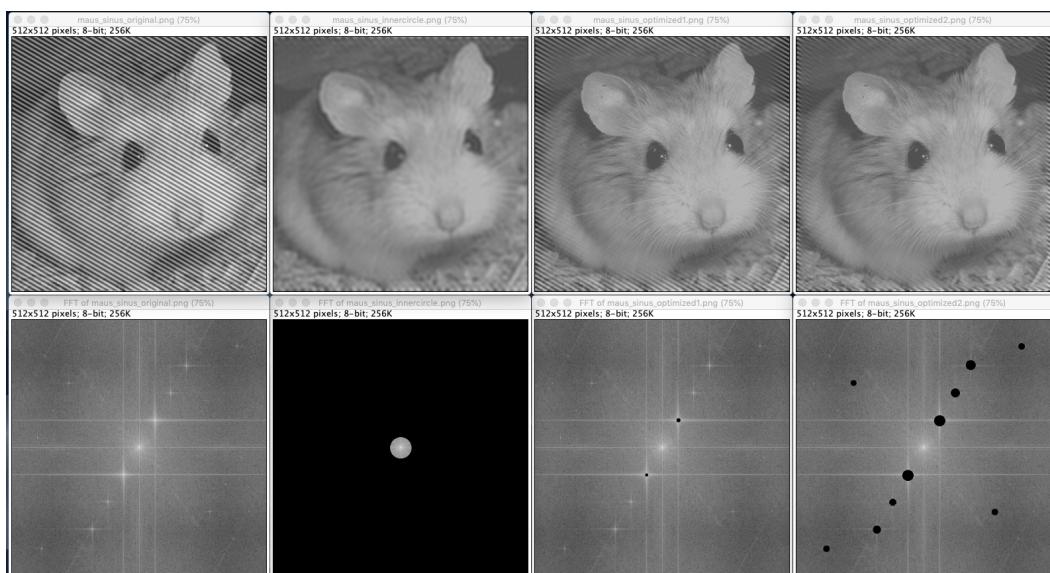


Abbildung 1.1: Maus mit Raster und darunter das jeweilige FF-transformierte Bild. Links das Original.

In dem Test-Bild *Maus* ist ersichtlich, dass eine Sinus-Welle eine Störung im Bild verursacht. Betrachtet man den Frequenzraum, so ist diese Welle durch mehrere sehr helle Stellen ersichtlich. Bei den Versuchen sie zu eliminieren fällt auf, dass es ausreicht, die zwei intensivsten Stellen zu entfernen (siehe Abb. 1.1 3. Bilderpaar von links), um einen starken Effekt zu erzielen. Lediglich die Randbereiche bleiben gleich.

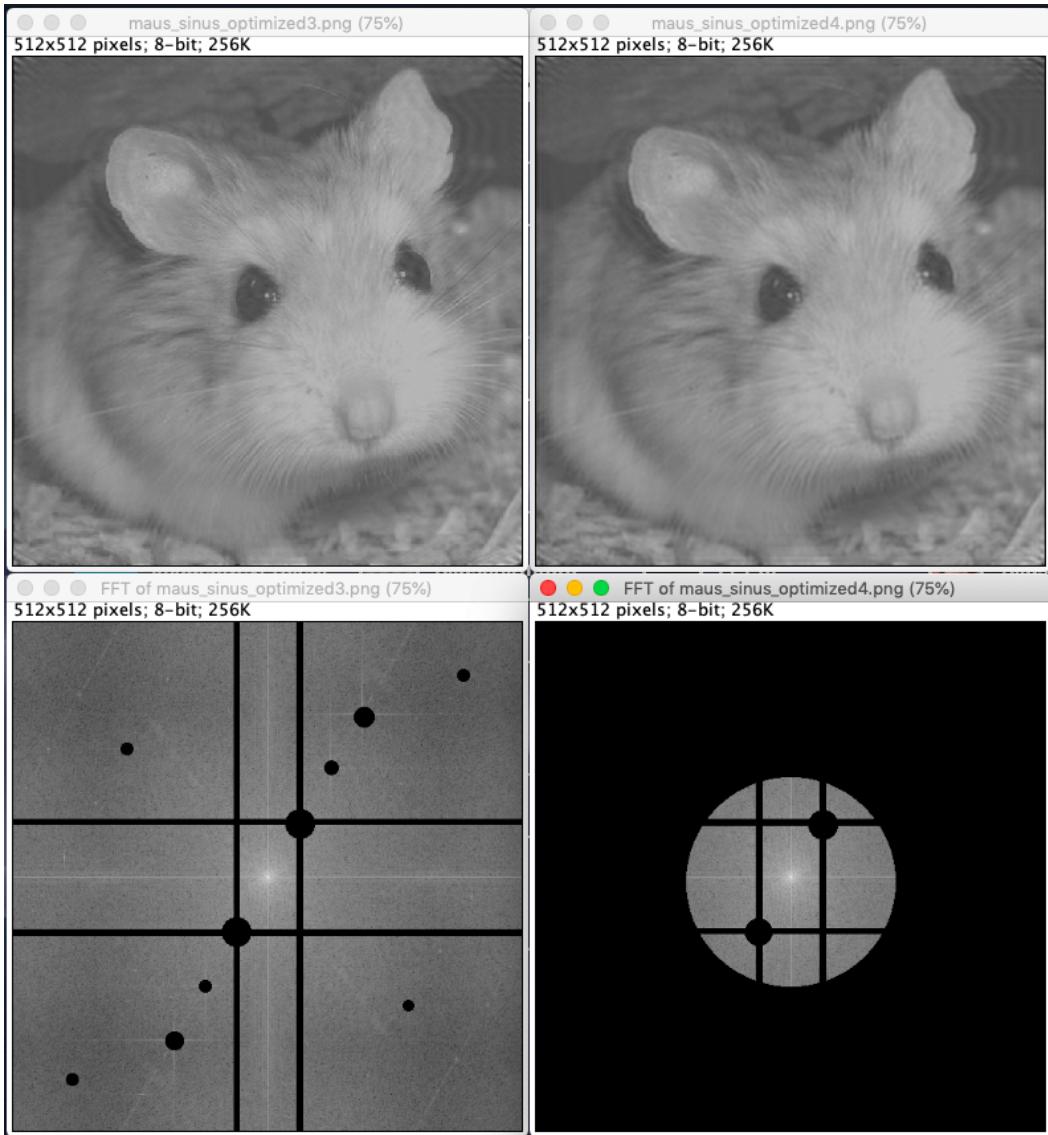


Abbildung 1.2: Weitere Experimente im Frequenzraum der Maus.

Bei weiterer Optimierung des Frequenzraum-Bildes kann eine Verbesserung in den Randbereichen festgestellt werden, allerdings wird bei der Eliminierung großflächiger Frequenzbereiche die Bildqualität schlechter. Weiteres lässt sich beobachten, dass im Bereich um die Kanten mehrere abgeschwächte, zuvor nicht vorhandene, Kanten auftreten (siehe Abb. 1.2).

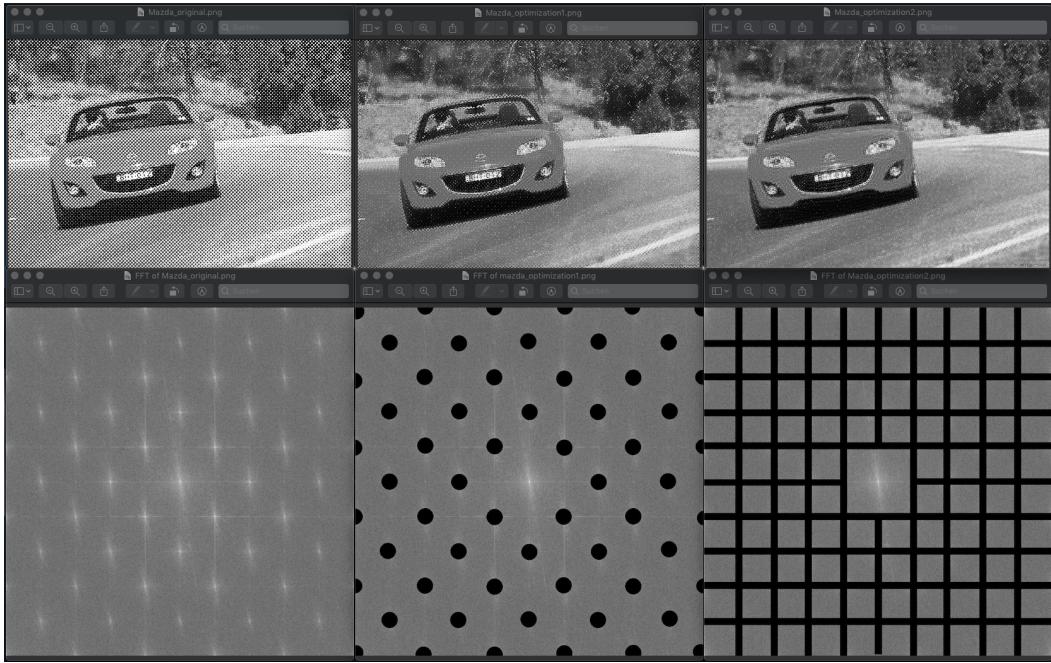


Abbildung 1.3: Bild eines Mazdas mit starkem gitterförmigen Rauschen.

In dem Test-Bild 1.3 ist ein gitterförmiges Raster, wie bei einem Siebdruck, zu erkennen, welches im Frequenzraum durch ein horizontales und vertikales Raster erkennbar ist. Bei punktförmiger Eliminierung im FR kann eine deutliche Verbesserung der Bildqualität festgestellt werden. So wird beispielsweise das Kennzeichen wieder lesbar (siehe Abb. 1.3). Werden allerdings auch die *Strahlen* der Punkte entfernt, so wird die Bildqualität wieder schlechter. Ähnlich zu dem Test-Bild 1.2 treten auch hier Probleme in den Randbereichen auf.

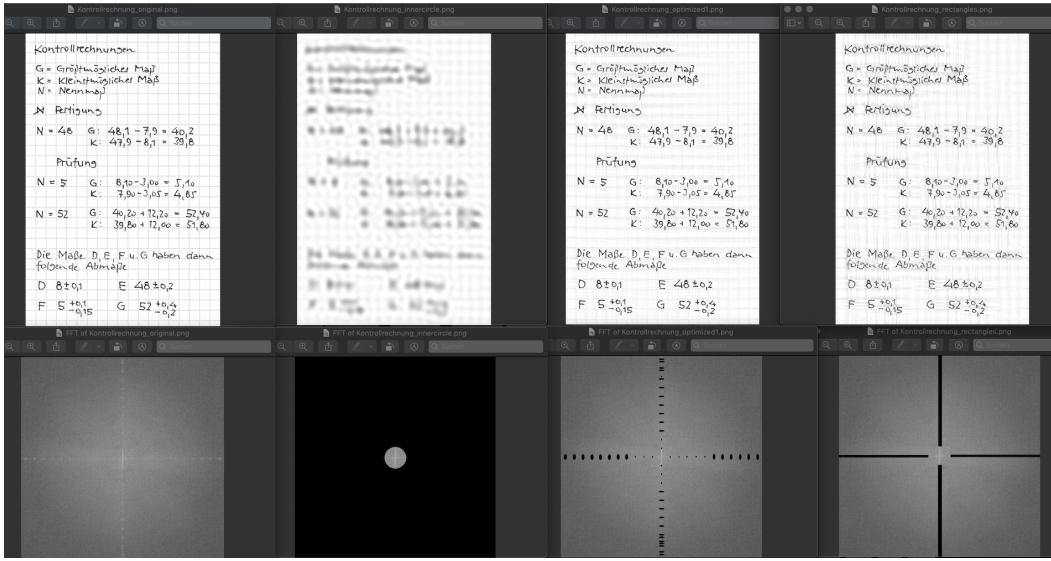


Abbildung 1.4: Handschriftliche Rechnung auf kariertem Papier. Es wird versucht das Karomuster des Papiers zu entfernen.

In dem Test-Bild 1.4 ist ein kariertes Hintergrund zu erkennen, welcher im Frequenzraum durch Punkte auf der horizontalen und vertikalen Linie des mittleren Kreuzes erkennbar ist. Selbst bei völliger Eliminierung der Linien, bleibt eine Kontur des Hintergrunds erkennbar.

Fazit

Quantitativ betrachtet werden bei jeglicher Eliminierung von Inhalten im Frequenzraum Inhalte des Bildes eliminiert und dadurch die Bildqualität verschlechtert. Tatsächlich werden aber bei Eliminierung der richtigen Stellen Details sichtbar, welche vorher durch das Raster verdeckt wurden. Eine vollständige Entfernung des Rasters, wird durch rein manuelle Manipulation nicht zur Gänze gelingen. Da der Frequenzraum symmetrisch ist, wäre ein besserer Ansatz, die gewünschten Stellen ebenfalls symmetrisch zu entfernen. Idee: Programm vergleicht FR-Bild ohne Raster mit FR-Bild mit Raster und entfernt alle Stellen die eine große Differenz haben (außer in der Mitte). Um Artefakte gleichmäßig zu entfernen, könnte ein Bandreject-Filter (da der Bandpass-Filter die Artefakte isoliert) oder Notch-Pass-Filter auf das Frequenzraumbild angewandt werden.

1.6 Anisotrope Diffusion

a) Implementierung - Lösungsidee

Es soll ein Glättungsfilter mit Kantenerhaltung implementiert werden, dazu wird der Pseudocode aus dem Foliensatz 2_signalverbesserung_v1, Folie 35 als Vorlage verwendet. Es wird der Algorithmus dahingehend programmiert, dass unterschiedlich viele Iterationen sowie auch Kappa-Werte vom Benutzer eingegeben werden können. Es werden pro Himmelsrichtung 3x3 Felder große Arrays erstellt, die den jeweiligen Gradienten

rausfiltern.

Innerhalb der Schleife werden dann pro Iteration 8 Nabla Bilder, wieder jeweils pro Himmelsrichtung, als Ergebnismatrizen von *ConvolutionFilter.convolveDouble* abgespeichert.

Im nächsten Schritt werden wieder 8 Bilder in als Koeffizientenmatrizen berechnet. Hier ist die Berechnung des Pseudocodes nicht deckungsgleich mit der Formel in den Folien, wir werden die Formel aus Folie 33 zur Berechnung heranziehen.

$$c(|\nabla f|) = e^{-\frac{1}{2} \left(\frac{|\nabla f|}{K} \right)^2} \quad (1.1)$$

Mit diesen 8 Bildern werden die neuen Pixel des gefilterten Bildes errechnet. Dazu werden nur die Indizes der Nabla Bilder und den dazugehörigen Koeffizienten multipliziert und dann aufaddiert. Da die Distanzgewichte in Summe ungleich 1 sind, muss ein Normfaktor die Summe noch normieren.

Source Code

```

1 if (kappa == 0) kappa = 0.000001; // prevent division by zero
2 int radius = 1;
3
4 // rotated arrays for column major order
5 double[][] hN    = new double[][]{{0.0, 0.0, 0.0}, {1.0, -1.0, 0.0}, {0.0, 0.0,
   0.0}};
6 double[][] hNW   = new double[][]{{1.0, 0.0, 0.0}, {0.0, -1.0, 0.0}, {0.0, 0.0,
   0.0}};
7 double[][] hW    = new double[][]{{0.0, 1.0, 0.0}, {0.0, -1.0, 0.0}, {0.0, 0.0,
   0.0}};
8 double[][] hSW   = new double[][]{{0.0, 0.0, 1.0}, {0.0, -1.0, 0.0}, {0.0, 0.0,
   0.0}};
9 double[][] hS    = new double[][]{{0.0, 0.0, 0.0}, {0.0, -1.0, 1.0}, {0.0, 0.0,
   0.0}};
10 double[][] hSE   = new double[][]{{0.0, 0.0, 0.0}, {0.0, -1.0, 0.0}, {0.0, 0.0,
   1.0}};
11 double[][] hE    = new double[][]{{0.0, 0.0, 0.0}, {0.0, -1.0, 0.0}, {0.0, 1.0,
   0.0}};
12 double[][] hNE   = new double[][]{{0.0, 0.0, 0.0}, {0.0, -1.0, 0.0}, {1.0, 0.0,
   0.0}};
13
14 for (int iter = 0; iter < n; iter++) {
15   // convolution for each compass gradient
16   double[][] nablaN  = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
   height, hN, radius);
17   double[][] nablaNW = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
   height, hNW, radius);
18   double[][] nablaW  = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
   height, hW, radius);
19   double[][] nablaSW = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
   height, hSW, radius);
20   double[][] nablaS  = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
   height, hS, radius);
21   double[][] nablaSE = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
   height, hSE, radius);

```

```

22 double[][] nablaE = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
23   height, hE, radius);
24 double[][] nablaNE = ConvolutionFilter.convolveDouble(indataArrayDbl, width,
25   height, hNE, radius);
26 double[][] cN = new double[width][height];
27 double[][] cNW = new double[width][height];
28 double[][] cW = new double[width][height];
29 double[][] cSW = new double[width][height];
30 double[][] cS = new double[width][height];
31 double[][] cSE = new double[width][height];
32 double[][] cE = new double[width][height];
33 double[][] cNE = new double[width][height];
34 // Calculation from slide 33 Anisotropic Diffusion V
35 for (int x = 0; x < width; x++) {
36   for (int y = 0; y < height; y++) {
37     cN[x][y] = Math.exp(-0.5 * (Math.pow((nablaN[x][y] / kappa), 2.0)));
38     cNW[x][y] = Math.exp(-0.5 * (Math.pow((nablaNW[x][y] / kappa), 2.0)));
39     cW[x][y] = Math.exp(-0.5 * (Math.pow((nablaW[x][y] / kappa), 2.0)));
40     cSW[x][y] = Math.exp(-0.5 * (Math.pow((nablaSW[x][y] / kappa), 2.0)));
41     cS[x][y] = Math.exp(-0.5 * (Math.pow((nablaS[x][y] / kappa), 2.0)));
42     cSE[x][y] = Math.exp(-0.5 * (Math.pow((nablaSE[x][y] / kappa), 2.0)));
43     cE[x][y] = Math.exp(-0.5 * (Math.pow((nablaE[x][y] / kappa), 2.0)));
44     cNE[x][y] = Math.exp(-0.5 * (Math.pow((nablaNE[x][y] / kappa), 2.0)));
45   }
46 }
47 double dw = 1.0 / Math.sqrt(2.0); // diagonale distance weight
48 double normFactor = 1.0 / (4 + 4 * dw); // 4 * 1 + 4 * sqrt(2) - sum of distances
49
50 for (int x = 0; x < width; x++) {
51   for (int y = 0; y < height; y++) {
52     double newVal =
53       // old pixel value
54       indataArrayDbl[x][y] + normFactor * (
55         // straight compass directions north, west, south, east
56         cN[x][y] * nablaN[x][y]
57         + cW[x][y] * nablaW[x][y]
58         + cS[x][y] * nablaS[x][y]
59         + cE[x][y] * nablaE[x][y]
60         // diagonal compass directions north-west, south-west, south-east, north-east
61         // multiplied with diagonal distance weight
62         + dw * cNW[x][y] * nablaNW[x][y]
63         + dw * cSW[x][y] * nablaSW[x][y]
64         + dw * cSE[x][y] * nablaSE[x][y]
65         + dw * cNE[x][y] * nablaNE[x][y]);
66
67     indataArrayDbl[x][y] = newVal;
68   }
69 }
70 }
71 ImageJUtility.showNewImage(indataArrayDbl, width, height,
72   "anisotropic diffusion, n=" + n + ", kappa=" + kappa);

```

b) Tests mit verschiedenen Bildern und unterschiedlichen Kappa

Es wurde bei den 4 Testbildern Kombinationen aus Kappa 5/50 und Iterationen 5/50 ausgeführt. Auffällig bei hohem Kappa ist der Detailverlust, eine hohe Anzahl an Iterationen hingegen scheint kaum Auswirkungen auf das Ergebnis zu haben.



Abbildung 1.5: Kameramann links als Original und rechts gefiltert (Iterationen 5, Kappa 5).



Abbildung 1.6: Kameramann links als Original und rechts gefiltert (Iterationen 5, Kappa 50).



Abbildung 1.7: Kameramann links als Original und rechts gefiltert (Iterationen 50, Kappa 5).

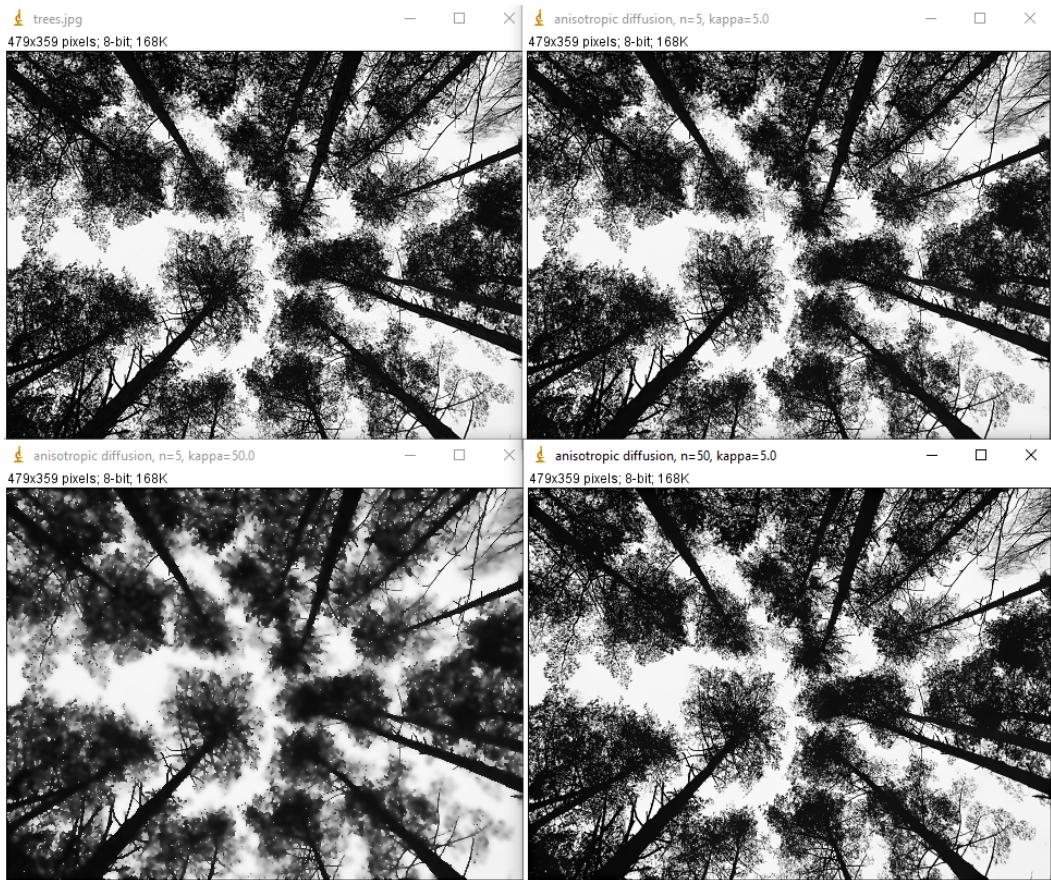


Abbildung 1.8: Bäume oben als Original und gefiltert mit den Parametern (Iterationen 5, Kappa 5), (Iterationen 5, Kappa 50), (Iterationen 50, Kappa 5).

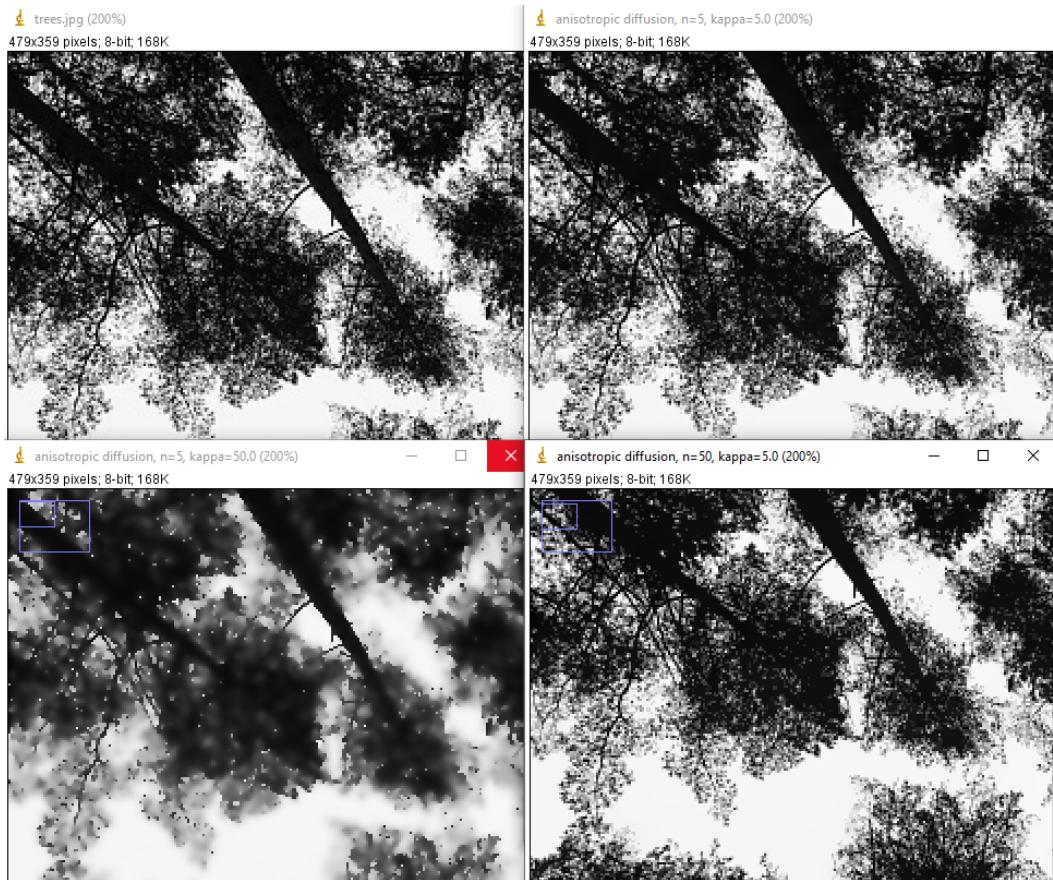


Abbildung 1.9: Bäume herangezoomt.

Bei den Blättern der Bäume ist der Detailverlust gut sichtbar, hier wirkt der Filter bereits wie ein Aquarellbild.

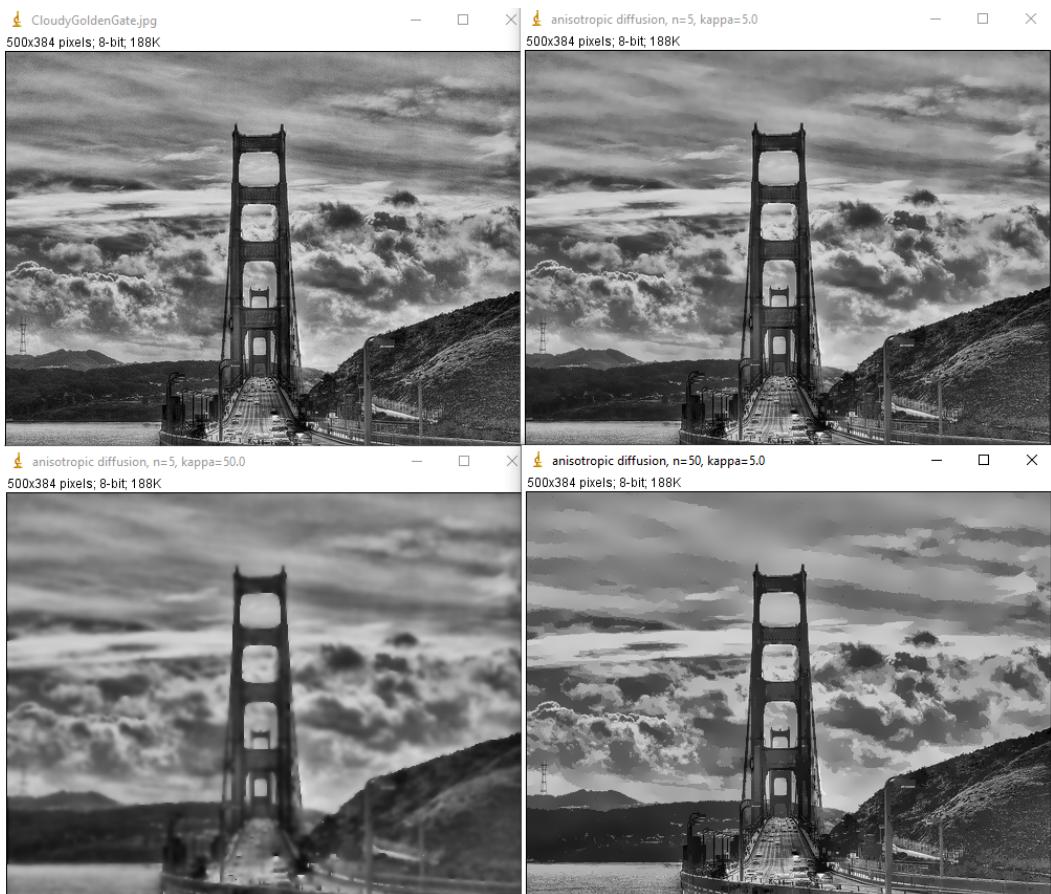


Abbildung 1.10: Brücke links oben als Original und gefiltert mit den Parametern (Iterationen 5, Kappa 5), (Iterationen 5, Kappa 50), (Iterationen 50, Kappa 5).



Abbildung 1.11: Brücke links als Original und rechts gefiltert (Iterationen 5, Kappa 5).



Abbildung 1.12: Brücke links als Original und rechts gefiltert (Iterationen 5, Kappa 50).

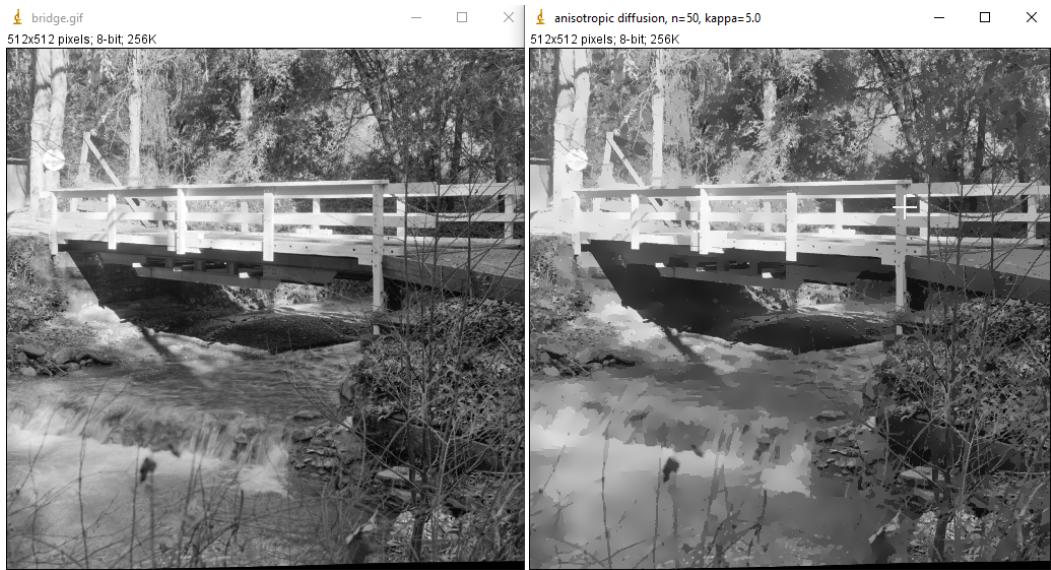


Abbildung 1.13: Brücke links als Original und rechts gefiltert (Iterationen 50, Kappa 5).

c) Teilaufgabe



Abbildung 1.14: Visualisierung der 8 Nabla Bilder angeordnet in den Himmelsrichtungen.

```

▼ cN = {double[487][1]@2096}
> 0 = {double[487]@2220}[4.170515773020078E-212, 0.6065306597126334, 0.19789869908361465, 1.0, 0.9231163463866358, 0.9801986733067553, 0.6065306597126334, 0...
> 1 = {double[487]@2221}[7.970880740974811E-215, 0.6065306597126334, 0.2780373004531941, 0.9801986733067553, 0.835270211411272, 0.9231163463866358, 0.7261490370736908, 0.835...
> 2 = {double[487]@2222}[1.4636966381778759E-217, 0.6065306597126334, 0.37531109885139957, 0.9231163463866358, 0.7261490370736908, 0.835270211411272, 0.7261490370736908, 0.835...
> 3 = {double[487]@2223}[1.4636966381778759E-217, 0.7261490370736908, 0.37531109885139957, 0.6065306597126334, 0.835270211411272, 0.7261490370736908, 0.835...
> 4 = {double[487]@2224}[7.970880740974811E-215, 0.9231163463866358, 0.37531109885139957, 0.4867522559599717, 0.7261490370736908, 0.6065306597126334, 0.923...
> 5 = {double[487]@2225}[4.170515773020078E-212, 0.9801986733067553, 0.2780373004531941, 0.37531109885139957, 0.7261490370736908, 0.4867522559599717, 0.980...
> 6 = {double[487]@2226}[1.0126079754914655E-206, 0.9801986733067553, 0.19789869908361465, 0.37531109885139957, 0.7261490370736908, 0.37531109885139957, 0...
> 7 = {double[487]@2227}[4.699043469073907E-204, 0.9231163463866358, 0.19789869908361465, 0.4867522559599717, 0.7261490370736908, 0.37531109885139957, 0.923...
> 8 = {double[487]@2228}[4.170515773020078E-212, 0.9801986733067553, 0.9801986733067553, 0.9801986733067553, 0.835270211411272, 0.92311...
> 9 = {double[487]@2229}[7.970880740974811E-215, 0.9801986733067553, 0.9801986733067553, 0.9801986733067553, 0.835270211411272, 0.9801986733067553, 0.83...
> 10 = {double[487]@2230}[1.4636966381778759E-217, 0.9801986733067553, 0.9801986733067553, 1.0, 0.9801986733067553, 0.9231163463866358, 0.9801986733067553, 0...
> 11 = {double[487]@2231}[4.377491037053051E-223, 0.9231163463866358, 1.0, 0.9231163463866358, 0.9231163463866358, 0.9801986733067553, 0...
> 12 = {double[487]@2232}[4.377491037053051E-223, 0.9801986733067553, 1.0, 1.0, 0.9231163463866358, 0.9231163463866358, 0.9801986733067553, 0.72614903707369...
> 13 = {double[487]@2233}[4.377491037053051E-223, 0.9801986733067553, 1.0, 1.0, 0.9231163463866358, 0.9231163463866358, 0.9801986733067553, 0.72614903707369...
> 14 = {double[487]@2234}[2.582403252784188E-220, 0.9801986733067553, 0.9801986733067553, 0.9801986733067553, 0.9801986733067553, 0.9231163463866358, 0.980...
> 15 = {double[487]@2235}[2.582403252784188E-220, 0.9231163463866358, 0.9801986733067553, 1.0, 0.9801986733067553, 0.9231163463866358, 0.9801986733067553, 0...
> 16 = {double[487]@2236}[2.582403252784188E-220, 0.9231163463866358, 1.0, 0.9801986733067553, 0.9801986733067553, 1.0, 0.9801986733067553, 1.0, 0.98019867330...
> 17 = {double[487]@2237}[2.582403252784188E-220, 0.9231163463866358, 1.0, 0.9801986733067553, 0.9801986733067553, 0.9801986733067553, 1.0, 1.0, 0.98019867330...
> 18 = {double[487]@2238}[4.377491037053051E-223, 0.9231163463866358, 0.9801986733067553, 0.9231163463866358, 1.0, 0.9801986733067553, 1.0, 1.0, 0.92311634638...
> 19 = {double[487]@2239}[4.377491037053051E-223, 0.9231163463866358, 1.0, 0.9801986733067553, 0.9801986733067553, 0.9231163463866358, 0.9801986733067553, 0...
> 20 = {double[487]@2240}[4.377491037053051E-223, 0.9231163463866358, 1.0, 0.9231163463866358, 1.0, 0.9801986733067553, 0.9231163463866358, 1.0, 0.98019867330...
> 21 = {double[487]@2241}[4.377491037053051E-223, 0.9801986733067553, 1.0, 0.9231163463866358, 1.0, 0.9231163463866358, 0.9231163463866358, 1.0, 0.83527021141...
> 22 = {double[487]@2242}[4.377491037053051E-223, 0.9801986733067553, 1.0, 0.9231163463866358, 0.9231163463866358, 0.9801986733067553, 1.0, 0.9231163463866358, 0.980198673306755...
> 23 = {double[487]@2243}[4.377491037053051E-223, 0.9801986733067553, 1.0, 0.9231163463866358, 0.9231163463866358, 0.9801986733067553, 1.0, 0.9231163463866358, 0.980198673306755...

```

Abbildung 1.15: Inhalt des c-Nord Bildes, visualisieren ist hier nicht zielführend, da die Werte beinahe schwarze Bilder erzeugen.

1.7 Stressanalyse

a) EKG

Das EKG (Elektrokardiogramm) gehört zu den wichtigsten medizinischen Untersuchungsmethoden und kann dabei helfen, Probleme am Herzen festzustellen, wie etwa Herzrhythmusstörungen. Der Sinusknoten, der sich an der rechten Vorhofwand des Herzens befindet, fungiert als Taktgeber indem er elektrische Impulse sendet, die sich dann über das gesamte Herz ausbreiten, damit es arbeiten kann. Das EKG misst genau diese Ströme, wie sie sich ausbilden und wieder zurück gehen und man kann ablesen, wie häufig und schnell das Herz schlägt (Stiftungs-Gesundheitswissen, 2022¹).

Ablauf eines Herzschlages

Ein Herzschlag kann mit dem Ablauf der P-Welle, QRS-Komplex und der T-Welle beschrieben werden, wie in Abb. 1.16 zu sehen.

¹<https://www.stiftung-gesundheitswissen.de/gesundes-leben/koerper-wissen/was-ist-ein-ekg>

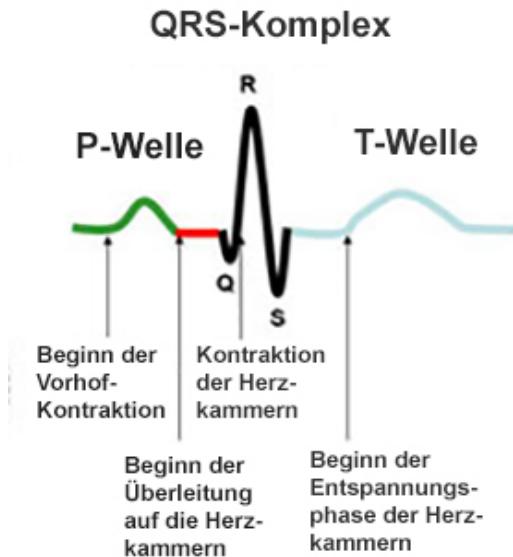


Abbildung 1.16: Der typische Ablauf eines Herzschlages (Kahr, 2019).

Wie in Abb. 1.16 zu sehen, beginnt der Herzschlag mit der P-Welle – sie entsteht durch die Erregung in den Vorhöfen. Da die Muskelmasse der Vorhöfe recht gering ist, ist auch die Kurve relativ klein. Danach zeigt sich eine isoelektrische Linie (PQ-Strecke), in der sich der Erregungszustand nicht ändert. Der QRS-Komplex zeigt die Erregungsausbreitung in den Kammern. Zuerst sieht man eine Depolarisation in Richtung der Ventrikelbasis – den Q-Zacken –, danach eine Depolarisation entlang der Herzachse – den R-Zacken – und anschließend eine Depolarisation der subepikardialen Anteile an der Basis des linken Ventrikels – der S-Zacken. Die T-Welle zeigt eine Erregungsrückbildung, wobei die Repolarisation in verkehrter Reihenfolge verläuft, also von außen nach innen. Wenn die T-Welle beginnt, ist bereits der größte Teil des Herzvolumens schon ausgeworfen. Bei hoher Herzfrequenz ist der QT-Strecke verkürzt, im Normalfall ist sie ca. 300 ms lang. Ab und an kann man nach einer T-Welle auch eine U-Welle beobachten, deren Funktion ist aber noch nicht genau erforscht (Krimmer, 2018²).

Herzvariabilität (Heart Rate Variability, kurz HRV):

Generell schlägt das Herz bei Ruhe langsamer als bei Belastung, wobei die Zahl der Schläge pro Minute als Herzfrequenz, manchmal auch als Puls bezeichnet wird. Die Herzschlagvariabilität zeigt eine Variabilität der Herzschlagfolge an. Je entspannter das Herz ist, desto variabler schlägt es. Eine hohe HRV zeigt sich vor allem bei gut trainierten Personen oder wenn sich jemand sehr wohl fühlt. Faktoren, die zu einer gesunkenen HRV führen können, sind unter anderem Stress, Schlafmangel oder hoch intensives Training ohne genügend Ruhepausen. Begünstigend für eine hohe HRV sind beispielsweise

²EKG-Kurve. URL: <https://m.thieme.de/viamedici/vorklinik-faecher-physiologie-1509/a/29339.htm>

Wohlbefinden, ausgewogenes Training oder Meditieren (Hottenrott, 2019³). Wie in Abb. 1.17 zu sehen, wird die HRV zwischen den R-Zacken gemessen.

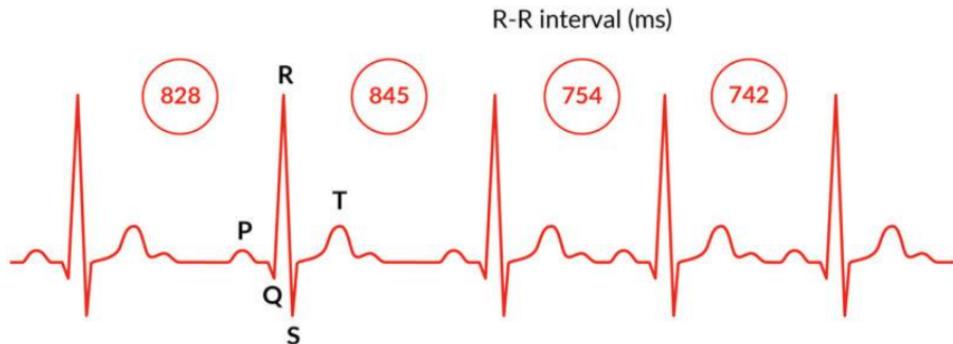


Abbildung 1.17: [HRV] Die HRV ist im Idealfall immer unterschiedlich – sie entspricht also den unterschiedlichen RR-Intervallen (Hoffman, o.J.).

Stress hat einen besonderen Einfluss auf die HRV, bzw. auf den Ruhepuls. Die HRV wird durch die sympathischen und parasympathischen Zweige autonome Nervensystem gesteuert und bei Stress ist die sympathische Seite des autonomen Nervensystems aktiviert – das System ist somit ihm „Kämpfe oder Fliehe“-Stadium. Der Körper ist also in einem aktiven Zustand und bereit, leistungsfähig zu sein. Es werden Stresshormone freigesetzt und diese wiederum bewirken eine schnellere Herzfrequenz und Blutdruck und verringert dabei gleichzeitig die HRV. Wenn der Stress vorbei ist, wird die parasympathische Seite aktiv – der Körper entspannt sich, die Herzfrequenz verringert sich und die HRV kann steigen. Durch das gute Zusammenspiel beider Seiten, kann sich unser Körper und das Herz den verschiedenen Situationen gut anpassen. Zirka 30% der HRV machen die Gene aus – den Rest kann der Mensch selber trainieren, etwa durch moderates Training, Achtsamkeitsübungen und Ernährung (Hoffman, o.J.⁴). Neben dem Fitnesszustand bzw. dem mentalen Zustand spielt auch das Alter eine große Rolle, denn je älter man wird, desto niedriger wird die HRV. Was die „idealen“ Werte sind, kann man jedoch nicht sagen, denn es gibt keine Grenz- oder Normwerte. Die Bewertung, ob HRV-Werte „gut“ oder „schlecht“ sind, sollte man deshalb immer nur im Vergleich mit den eigenen, vorangegangenen Werten treffen und im Idealfall immer zur gleichen Zeit messen (Lenertz, 2021⁵).

b) Datensatz Analyse

Analysiert wurde ein EKG mit Bewegung und eines ohne Bewegung, hauptsächlich im Tool „MS Excel“ und Matlab.

³<https://www.polar.com/blog/de/herzfrequenzvariabilitaet-was-du-wissen-musst-und-warum-sie-fuer-dein-training-nuetzlich-ist>

⁴<https://www.firstbeat.com/de/blog-de/was-ist-die-herzratenvariabilitaet-hrv-und-wieso-ist-sie-wichtig>

⁵<https://www.runnersworld.de/training-basiswissen/was-ist-eigentlich-die-herzfrequenzvariabilitaet>

EKG mit Bewegung

Ein Belastungs-EKG bedeutet, dass die EKG-Messung durchgeführt wird, während der Patient einer körperlichen Belastung ausgesetzt ist, etwa Radfahren oder Gehen. Die Formel für die maximale Herzfrequenz lautet dabei: 220 minus Lebensalter. Ein Belastungs-EKG wird auch nur als bedingt aussagekräftig gesehen, es wird in der Regel nur 85% der maximalen Herzschlagfrequenz als Zielwert angesetzt (Rosenbaum-Medani & Seidl-Konzett, o.J.⁶).

Bei dem Datasheet des EKGs mit Bewegung konnte man herauslesen, dass es 1000 Abtastungen pro Sekunde gab und es insgesamt 32701 Messdaten vorlagen.

EKG ohne Bewegung

Das Ruhe-EKG wird normal im Liegen gemacht und sollte immer vor einem Belastungs-EKG durchgeführt werden, da im Ruhe-EKG die Normalwerte sichtbar werden. Es gibt EKGs mit unterschiedlich vielen Ableitungen, je nachdem wie viele Elektroden verwendet und diese platziert werden.

Im Datensatz waren 31049 Datensätze und es wurden wieder 1000 Abtastungen pro Sekunde gemacht.

Baseline-Korrektur

Zuerst war eine Baseline-Korrektur gefragt. Hierfür wurde ein Mittelwert generiert – in dem Fall von den ersten 256 Werten – und dieser wurde von den Heartrates abgezogen. Somit liegen alle Daten auf der X-Achse, wie in Abb. 1.18 und Abb. 1.19 zu sehen.

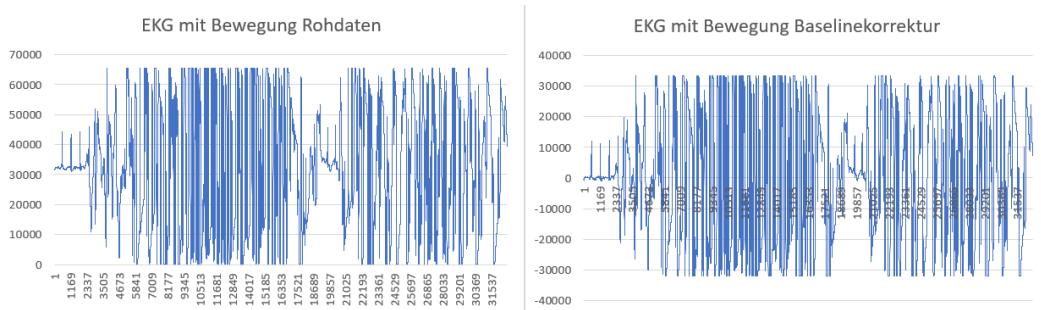


Abbildung 1.18: Die Daten befinden sich nun auf der 0-Linie , wodurch eine Auswertung erleichtert wird.

⁶<https://www.theaurora.at/leistungen/belastungs-ekg>

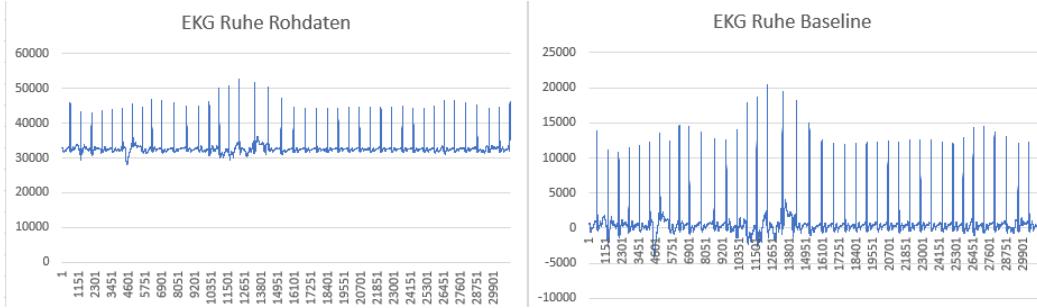


Abbildung 1.19: Die Daten sind nun auf der x-Achse ausgerichtet.

Glätten - EKG mit Bewegung

Das EKG mit Bewegung hat einen hohen Rauschanteil – einerseits durch das „natürliche“ 50hz – Brummen und andererseits natürlich durch die Bewegung selbst. Es wurde zunächst versucht, mit einem „üblichen“ Tiefpassfilter zu glätten – dem gleitenden Mittelwert – der jedoch nicht genug gefiltert hat. Somit wurde auf das Programm Matlab ausgewichen, wo die Daten mit dem „Gaussian-weighted moving average filter“ geglättet wurde. Gewählt wurde ein Fenster von 370, um eine möglichst effiziente Glättung zu bekommen, wie in Abb. 1.20 zu sehen.

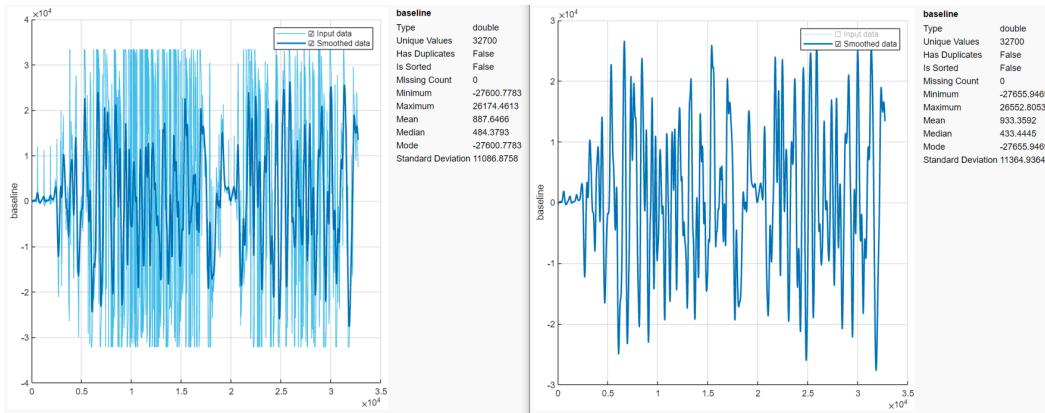


Abbildung 1.20: Die hellblauen Linien zeigen die Input-Daten, die dunkelblaue die gefilterten Daten. Rechts ist das rein gefilterte EKG zu sehen.

Glätten - EKG ohne Bewegung

Bei diesem EKG wurden 2 Filter ausprobiert in Excel. Einmal der „Gleitende Mittelwert“ und dann mit einem „normalen“ Tiefpassfilter 1. Ordnung. Die Glättung reicht eigentlich aus, da das EKG sehr schöne PQRST-Wellen hat und man auch die RR-Intervalle problemlos identifizieren kann, wie man in Abb. 1.21 erkennen kann.

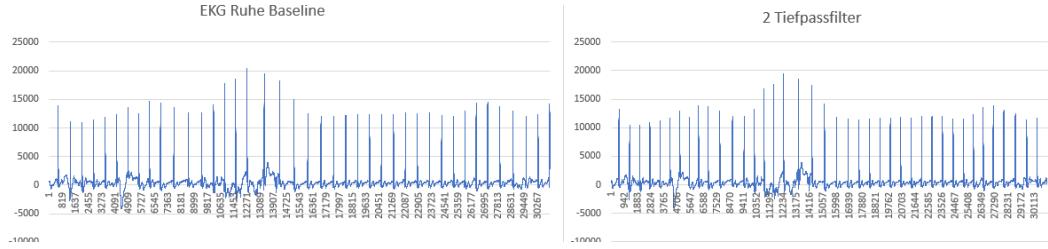


Abbildung 1.21: Links die Baseline-Korrektur, daneben das leicht gefilterte Bild, bei dem man die einzelnen Wellen gut erkennen kann.

Dennoch haben wir uns dazu entschlossen, auch einen „Gaussian-weighted moving average filter“ zu benutzen, wie man in Abb. 1.22 sehen kann.

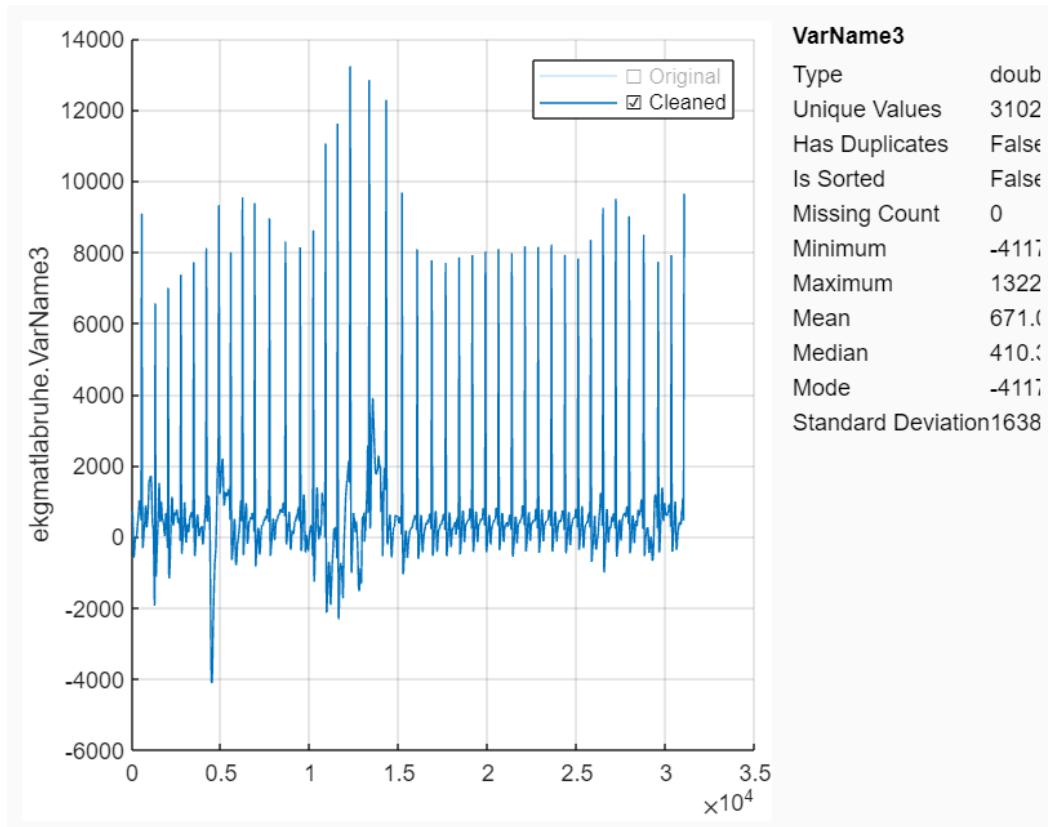


Abbildung 1.22: Man kann gut erkennen, dass der Filter nochmals vieles an Rauschen herausnimmt.

RR-Intervalle

Der RR-Abstand markiert die Dauer einer elektrischen Herzaktion und wird von R-Zacken zu R-Zacken gemessen.

EKG mit Bewegung

In Excel wurden dann die RR-Intervalle bestimmt. Aufgrund der Filterung sind viele RR-Intervalle größer, als im ursprünglichen Datenblatt und von der Anzahl verringert. Es zeigt sich, dass es mehrere r' – Spitzen gibt (R-Zacken, die gleich nach einem anderen R-Zacken kommen, ohne wirklich vorher in das S zu gehen, und sind vollkommen natürlich). Man kann noch erkennen, dass die Variabilität zwischen den Intervallen nicht sehr groß ist, relativ zu den Messdaten. Bei dem EKG mit Bewegung waren nach der Glättung noch 36 RR-Intervalle zu erkennen, die unterschiedlich groß waren. Die Herzfrequenz wird oft mit dem Puls gleichgesetzt, was aber nicht ganz korrekt ist, da der Puls die Ausdehnung und Kontraktion der Gefäßwände, die durch die Herzschläge verursacht werden, angibt (Fuchs, 2021⁷). Laut Mainertz (2022)⁸ entspricht der Puls meistens der Herzfrequenz, weshalb hier beide synonym gesehen werden. Die Herzfrequenz (Puls) aus dem EKG ist die Anzahl der Herzaktionen während einer bestimmten Zeiteinheit (flexikon, 2020b).

Wenn der Körper unter Belastung steht, schlägt das Herz schneller, das gleiche gilt auch bei innerer Anspannung. Sowohl bei Verliebtheit wie auch bei Stress, schlägt das Herz schneller. Durch den schnelleren Herzschlag pumpt der Körper mehr Blut durch die Adern, woraufhin der Puls steigt (Waitz & Peschel, 2022⁹)

Die Herzfrequenz ergibt sich durch die RR-Intervalle: 60/RR-Abstand in Sek. Bei einem Intervall von 314 ergibt sich eine Frequenz/Puls von 191 Schlägen pro Sekunde – das wäre schon ungesund, ist aber der Glättung geschuldet.

EKG ohne Bewegung

Die Herzfrequenz bei gesunden Menschen in Ruhe sollte bei 60-80 Herzschlägen pro Minute liegen und gibt den Normalzustand an (Waitz & Peschel, 2022). Die RR-Intervalle liegen alle recht gleichmäßig verteilt zwischen 600 und 800 ms, nur um den Datenbereich wo es die scheinbare Bewegung oder ein Husten/Räuspern des Probanden gab sind die Werte höher (ca. 1000 bzw 900). Der Puls, bzw. die Herzfrequenz liegt bei knapp über 80, das ist ein schöner Ruhepuls.

Die beste Lösung

um die RR-Intervalle zu identifizieren, wäre durch eine Wavelet-Transformation gewesen, die in Excel leider nicht umsetzbar ist und in Matlab fehlte die Kenntnis, wie das umsetzbar gewesen wäre. Es gibt für Matlab z.B. Algorithmen wie *R Wave Detection in the ECG* oder den Pan–Tompkins Algorithm. Da jedoch überall Anpassung vorzunehmen gewesen wären, die den Source-Code betroffen haben, konnten diese leider nicht eingesetzt werden.

⁷<https://de.beatayesterday.org/health/body-soul/puls-messen-5-methoden-deine-herzfrequenz-zu-ermitteln>

⁸<https://www.herzstiftung.de/ihre-herzgesundheit/das-herz/welcher-puls-ist-normal>

⁹<https://www.barmer.de/gesundheit-verstehen/bewegung-und-fitness/ruhepuls-1071436>

Herzzyklus

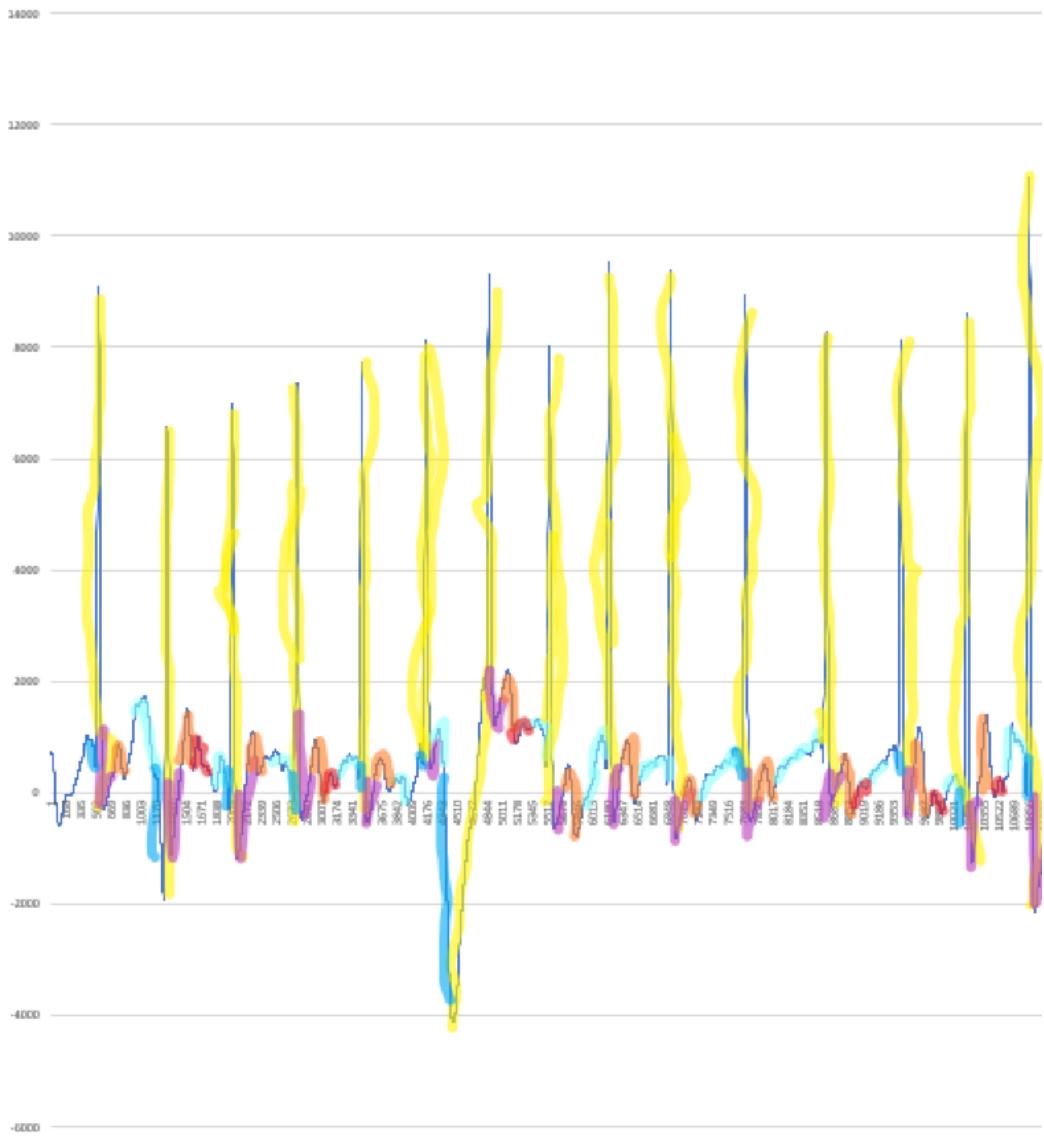


Abbildung 1.23: Hier sieht man die Herzzyklen im Ruhepuls.

In der Abb. 1.23 sieht man die unterschiedlichen Wellen, die einen Herzzyklus ausmachen. Türkis stellen die T-Wellen dar, blau den Q-Zacken, gelb den R-Zacken und lila den S-Zacken. Danach kommt die T-Welle in Orange und ab und an sieht man eine U-Welle, die ist rot eingezeichnet. Bei den vorliegenden Daten dauert bei einem ausgewählten Herzzyklus die P-Welle 111ms, der Q-Zacken 80ms, R-Zacken 2ms, der S-Zacken 102ms und die T-Welle 96ms. Ein Herzzyklus dauert somit (Datenbereich

von 19686 bis 20103) insgesamt 517ms, wobei davon der QT-Abstand 306ms sind. Der korrigierte QT-Abstand, der in diesem Herzzyklus liegt, beträgt mit der Bazett-Formel $QTc = QT - Zeit / \sqrt{RR - Abstand}$ (flexikon, 2020a) 11,24ms. Den Daten nach liegt ein Sinusrythmus vor und da der RQS-Komplex positiv in der Ableitung ist (Einthoven), scheint der Patient ein Indifferenztyp zu sein, das bei Erwachsenen physiologisch ist.

c) Herzvariabilität

Die Variation der Herzschlagintervalle in Abhängigkeit von der Zeit nennt man HRV (Beckmann, 2019¹⁰). Es wird das Mittlere Quadrat der sukzessiven Differenzen, das so genannte MQSD, als Parameter der Herzfrequenzvariabilität angenommen, wobei das MQSD aus dem Quadrat der Differenz aus den Momentanfrequenzen (angegeben in Schlägen/Min) zweier aufeinanderfolgender RR-Abstände errechnet wird, wobei die Quadrate addiert und durch die Zahl der Differenzen geteilt werden und anschließend die Wurzel gezogen wird.

EKG ohne Bewegung

Der MQSD, in der Einheit pro Minute, beträgt 0,00097333.

Zeitbereiche

SDNN: Anpassungsfähigkeit des Körpers. Die SDNN wird durch die Standardabweichungen aller gemessenen RR-Intervalle berechnet und ergibt 88,71017892ms, was einer mittelmäßigen SDNN entspricht. Werte unter 50ms wären ein Indikator, dass der Patient ein erhöhtes Sterberisiko haben könnte (Gricksch, 2019¹¹).

RMSD: Ein Messwert für den Einfluss des Parasympathikus auf die Herzschlagrate, weshalb der RMSD als Hinweisgeber für Erholungsfähigkeit gesehen wird. Mathematisch wird die RMSD als Quadratwurzel des Mittelwerts der Summe aller quadrierten Differenzen zwischen benachbarten RR-Intervallen beschrieben. Der RMSD ergibt einen Wert von 88,69. Es gibt jedoch keine Richtwerte, bei Langzeitmessungen gilt ein RMSD von für den RMSD 27 ± 12 ms bei Langzeitmessungen als Norm. In Kurzzeitmessungen ist es nicht wirklich aussagekräftig. Da es aber ein HRV-Standard ist, sollte er hier nicht fehlen.

Poincaré-Diagramm: Das Poincaré-Diagramm gilt als Hilfsmittel, um die Unregelmäßigkeit des Ruhepulses zu veranschaulichen und Kennwerte seiner Variabilität zu berechnen. Mathematisch ist dabei wie folgt vorzugehen: jedes RR-Intervall wird als Funktion des jeweils vorherigen RR-Intervalls dargestellt, das bedeutet s werden immer zwei aufeinander folgende RR-Intervalle einander zugeordnet und ins Diagramm eingetragen. Das heißt, dass der Wert eines RR-Intervalls (RRn) auf der x-Achse und der Wert des nächsten RR-Intervalls (RRn+1) auf der y-Achse gesucht werden und dementsprechend ein Punkt ins Diagramm eingetragen wird, wie in Abb. 1.24 zu sehen.

¹⁰<https://d-nb.info/1206538422/34>

¹¹<https://xn-hrv-herzratenvariabilitett-dcc.de/2019/01/berechnung-des-hrv-werts-sdnn>

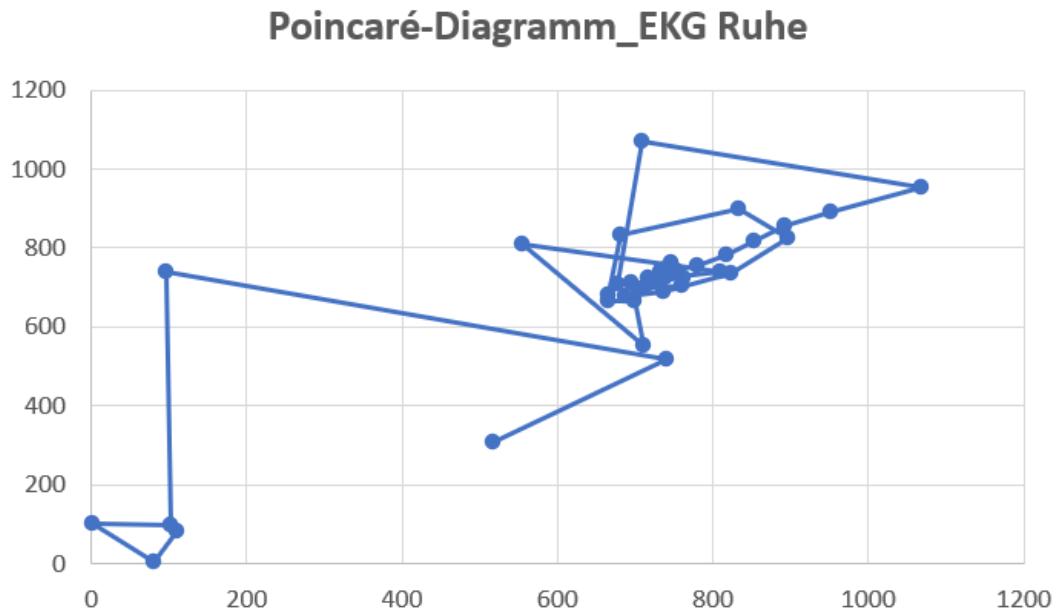


Abbildung 1.24: Iterative Darstellung der RR-Intervalle als nten Werte in Abhängigkeit von n-1. Man sieht, dass es 9 Abweichungen/Unregelmäßigkeiten gibt.

Fazit

Die HRV des Patienten ist gut, die Variabilität ist gegeben und der SDNN ist im normalen Bereich.

EKG mit Bewegung

Der MQSD beträgt 18,4249475.

SDNN: Die SDNN beträgt in diesem Datensatz 410, 35ms. Hier würde es auf eine starke Anpassungsfähigkeit schließen lassen.

RMSD: Die Erholungsfähigkeit liegt bei dem Wert 449,21. Da es keine wirklich Richtwerte gibt, kann man das nicht wirklich einordnen. Aufgrund der Datenglättung sollte man hier grundsätzlich vorsichtig sein.

Poincaré-Diagramm: Das Poincaré-Diagramm gibt wieder die Abhängigkeiten der Intervalle bildhaft aus. Wie in Abb. 1.25 zu sehen, liegen die RR-Intervalle verstreuter als bei dem EKG mit Ruhe. Eigentlich wird das Poincaré-Diagramm für den Ruhepuls benutzt, um dort Abweichungen zu finden, aber dem Interesse halber wurde es auch für den Datensatz mit Bewegung gemacht.

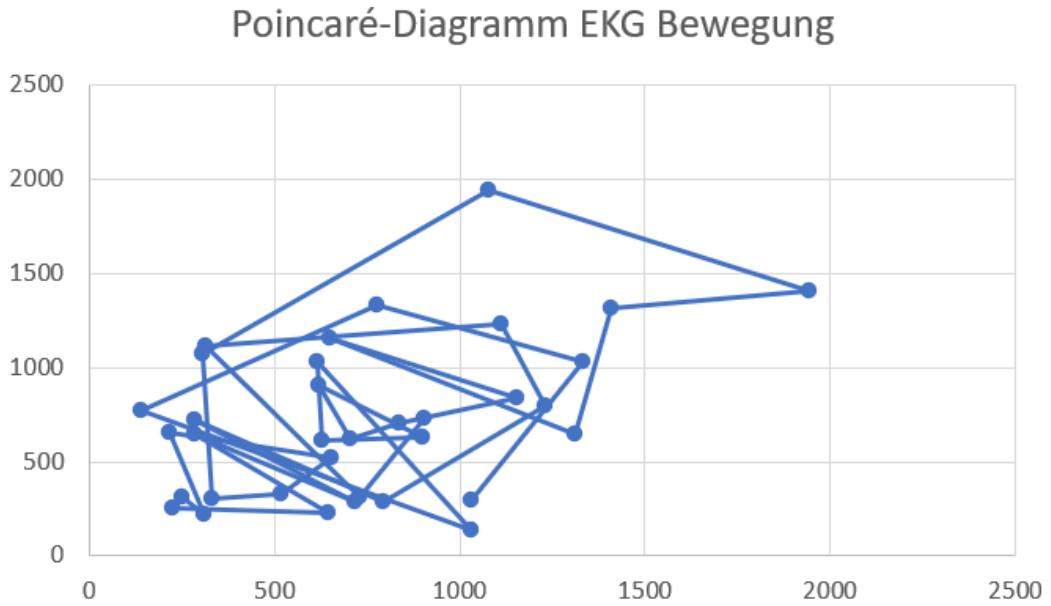


Abbildung 1.25: Die RR-Intervalle liegen breit verstreut herum, das Diagramm besteht quasi nur aus Abweichungen.

1.8 RL-Deconvolution

a) Implementierung - Lösungsidee

Aufgrund der am Übungszettel vorgegebenen Methodensignatur

```
1 private double[][] RDL(double[][] original, double[][] kernel, int width, int height
, int radius, int iterations)
```

wird die Richardson-Lucy implementiert. Die Formalparameter geben viel Aufschluss über die Funktionalität des Methodenkörpers. Es wird angenommen, dass *original* das verschwommene Eingangsbild ist, dass es zu rekonstruieren gilt. Weiters wird ein Kernel zur Faltung übergeben, wir wählen hier den *Mean-Kernel* mit beliebigen Radius, als auch den *Gauss-Kernel* mit Radius 1. Beide Kernel sind symmetrisch und brauchen somit kein Flip zu vollziehen. Da ein Parameter *iterations* vorhanden ist, werden auch die Schleifeniterationen in dieser Methode zu finden sein. Als Vorlage zur Implementierung dient Folie 14 aus dem Foliensatz 2_adaptiveFilter_v1. Die Variablen werden gleich benannt wie in der Folie, damit hier keine Fehler passieren und zur besseren Kontrolle. Da die Methode aber kein Guess Image entgegennimmt, wird dieses erst innerhalb des Methodenkörpers importiert.

Source Code

In Zeile 3 kann gewählt werden, ob man ein Bild als initiales Guess wählen möchte, wenn nicht, dann wird ein Bild mit dem Grauwert 128 oder mit zufälligen Double Werten befüllt. Ab Zeile 30 folgt der eigentliche RLD Algorithmus.

```

1 private double[][] RDL(double[][] original, double[][] kernel, int width, int height
, int radius, int iterations) {
2     double[][] guess = new double[width][height];
3     boolean guessIsAnImage = false;
4
5     if (guessIsAnImage) {
6         // import lena as guess
7         // import lena as guess
8         ImagePlus imp = new ImagePlus("C:\\\\path\\\\lena-8bit.png");
9         ImageProcessor ip = imp.getProcessor();
10        byte[] pixels = (byte[])ip.getPixels();
11        int widthGuess = ip.getWidth();
12        int heightGuess = ip.getHeight();
13        int[][] inDataArrIn = ImageJUtility.convertFrom1DByteArr(pixels, widthGuess,
heightGuess);
14        double[][] inDataArrayDbl = ImageJUtility.convert.ToDoubleArr2D(inDataArrIn,
widthGuess, heightGuess);
15        guess = inDataArrayDbl;
16    } else {
17        // create grey image as guess
18        for (int i = 0; i < width; i++) {
19            for (int j = 0; j < height; j++) {
20                guess[i][j] = 128.0;
21                // guess[i][j] = random.nextDouble() % 255;
22                // guess[i][j] = original[i][j];
23            }
24        }
25    }
26
27    double[][] mt = new double[width][height];
28    for (int iter = 0; iter < iterations; iter++) {
29        // init -  $c^t = h * g^t$ 
30        double[][] ct = ConvolutionFilter.convolveDoubleNorm(guess, width, height,
kernel, radius);
31        // start new iteration
32        for (int i = 0; i < width; i++) {
33            for (int j = 0; j < height; j++) {
34                // prevent division by zero
35                double divisor = ct[i][j] == 0.0 ? 0.0001 : ct[i][j];
36                // calculate the quotient between original and new guess
37                mt[i][j] = original[i][j] / divisor;
38            }
39        }
40        // update guess for next iteration :  $g^{(t+1)} = g^t \cdot (m^t * h)$ 
41        double[][] mth = ConvolutionFilter.convolveDoubleNorm(mt, width, height, kernel,
radius);
42        for (int i = 0; i < width; i++) {
43            for (int j = 0; j < height; j++) {
44                double res = guess[i][j] * mth[i][j];
45                // correct black artefacts caused by value overrun
46                if (res > 255.0) {
47                    // System.out.println("x=" + i + " y=" + j + " val=" + res);
48                    guess[i][j] = 255.0;
49                } else {
50                    guess[i][j] = guess[i][j] * mth[i][j];
51                }
52            }
}

```

```
53     }
54 }
55 return guess;
56 }
```

Tests

Unser eindeutigstes Ergebnis bekamen wir indem wir den Kameramann mit unserem selbstgebauten *Mean-Filter und Radius=3* unscharf gemacht haben und RDL ebenfalls mit *Radius=3* wieder scharf gestellt haben (siehe Abb.1.26). Kanten sind wieder scharf und die Details im Hintergrund tauchen wieder auf. Es fiel aber auf, dass in den Weißbereichen schwarze Artefakte auftauchen (siehe Abb.1.27), wir korrigieren das, indem wir jeden Wert größer 255.0 auf 255.0 setzen (siehe Abb.1.28).



Abbildung 1.26: l.o. unscharf gemacht mit Mean-Filter, r.o. RLD mit 10 Iterationen, l.u. 30 Iterationen und r.u. 50 Iterationen.

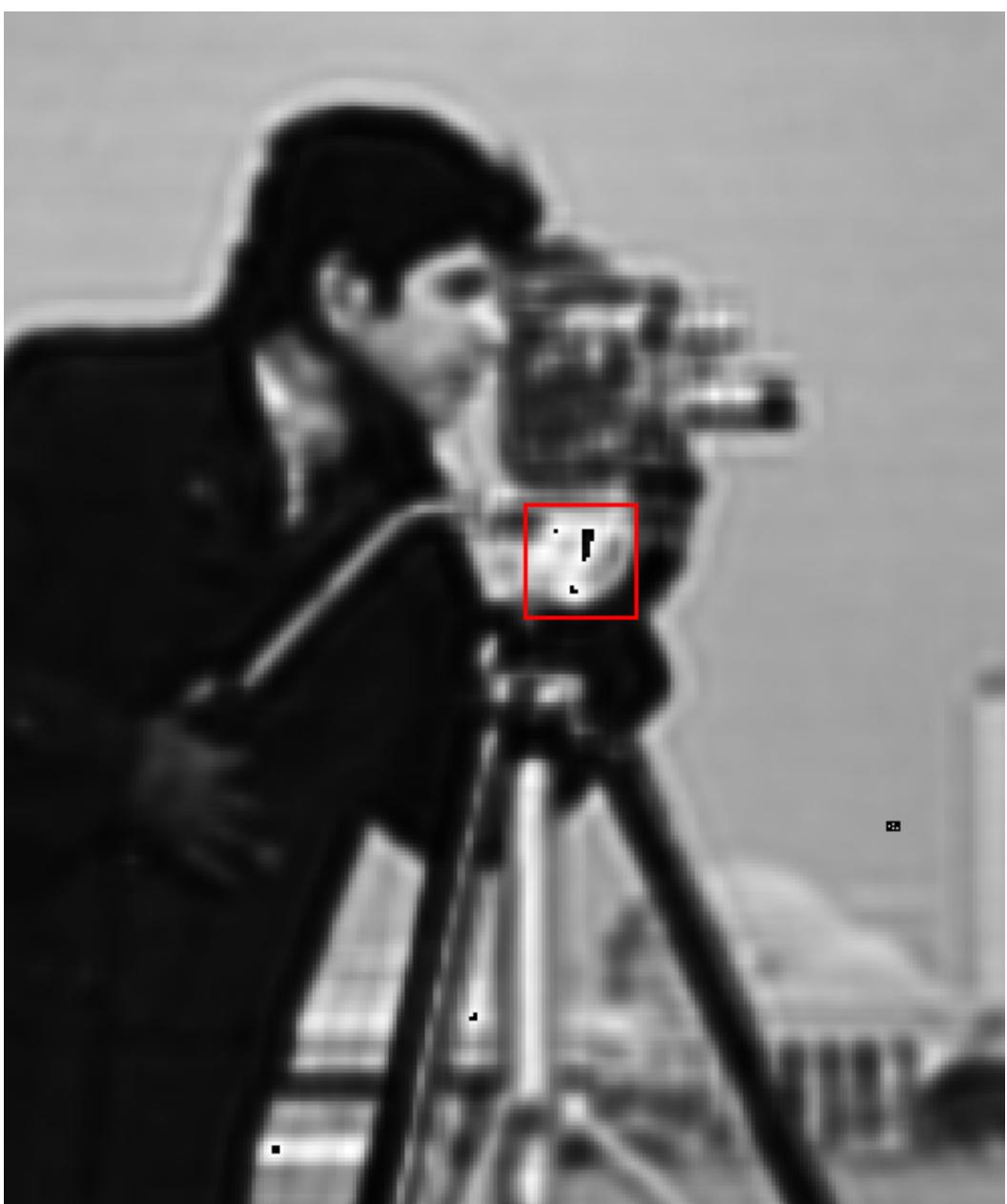


Abbildung 1.27: Fehlerhafte Artefakte innerhalb von weißen Bereichen.



Abbildung 1.28: Korrigierter Durchlauf. Auch bei 200 Iterationen tauchen nun keine Artefakte mehr auf.

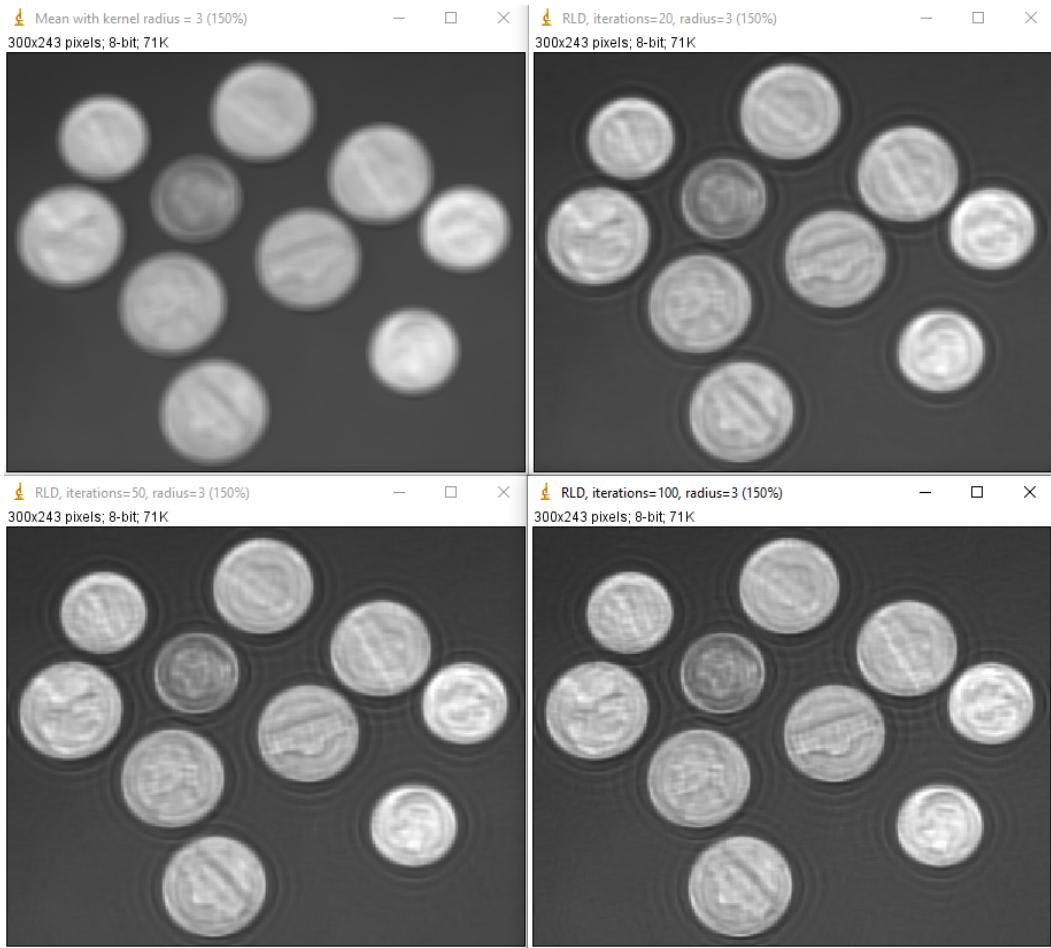


Abbildung 1.29: Vergleichsweise anwenden einer großen Filtermaske und einer kleinen die mehrmals hintereinander angewandt wurde.

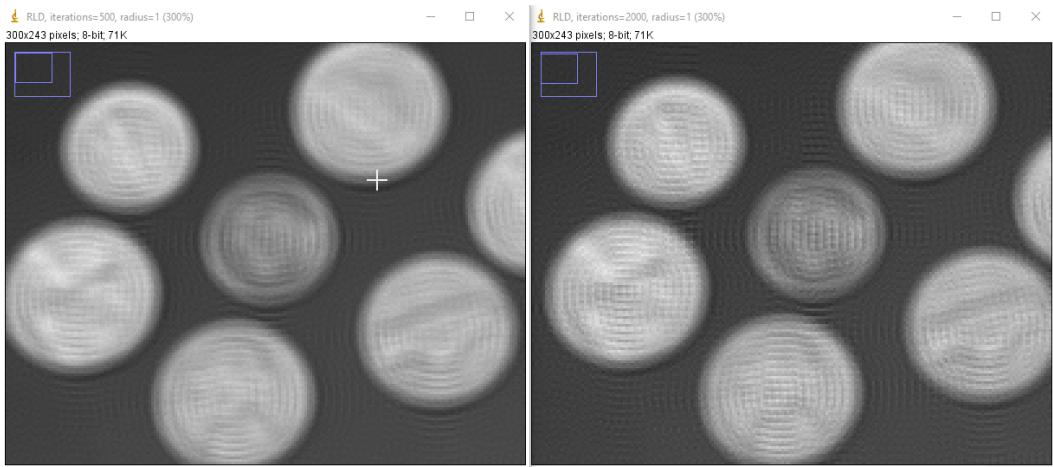


Abbildung 1.30: Vergleichsweises anwenden einer großen Filtermaske und einer kleinen die mehrmals hintereinander angewandt wurde.

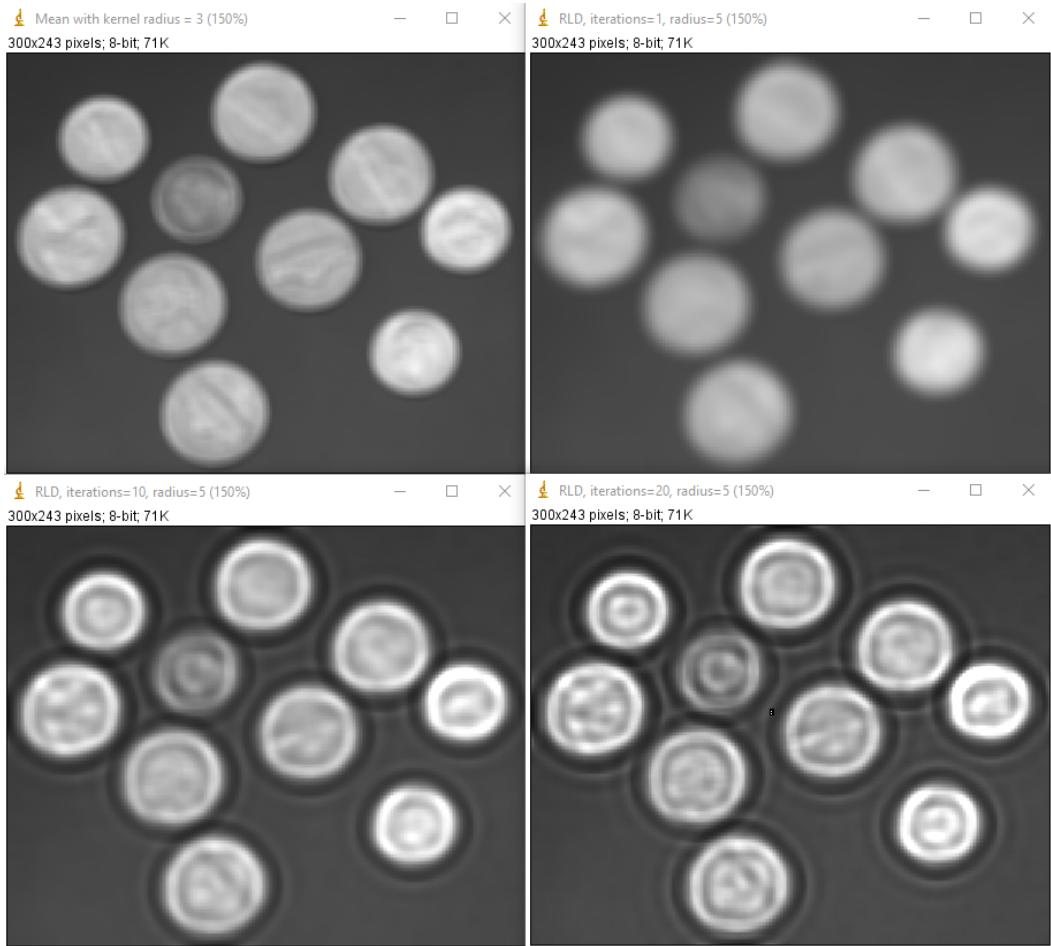


Abbildung 1.31: Vergleichsweises anwenden einer großen Filtermaske und einer kleinen die mehrmals hintereinander angewandt wurde.

b) Tests mit einem Bild (Lena) und einem zufälligen Bild als g0

Die Kanten des initialen g0-Bildes bleiben sehr lange erhalten, hier können die besseren Ergebnisse mit kleinen Radien erzielt werden. Die besten Resultate erzielen wir mit einem monotonen Bild als g0.



Abbildung 1.32: Graustufen Lena als g_0 (guess-Bild) nach 50 Iterationen immer noch die Kanten von Lena ersichtlich.



Abbildung 1.33: Mit 300 Iterationen immer noch leicht erkennbar.



Abbildung 1.34: Mit 50 und 300 Iterationen und Radius 3 mit einem zufällig generiertem Bild als g_0 .



Abbildung 1.35: Mit 50 und 300 Iterationen und Radius 1 mit einem zufällig generiertem Bild als g_0 .

Beim Testbild Brücke (Abb. 1.36) wurde das Originalbild mit fünffachen Smooth unscharf gemacht und danach mit drei unterschiedlichen Parametern der RLD Filter angewandt. Das beste Ergebnis lieferte $Radius=2$ mit 100 Iterationen.

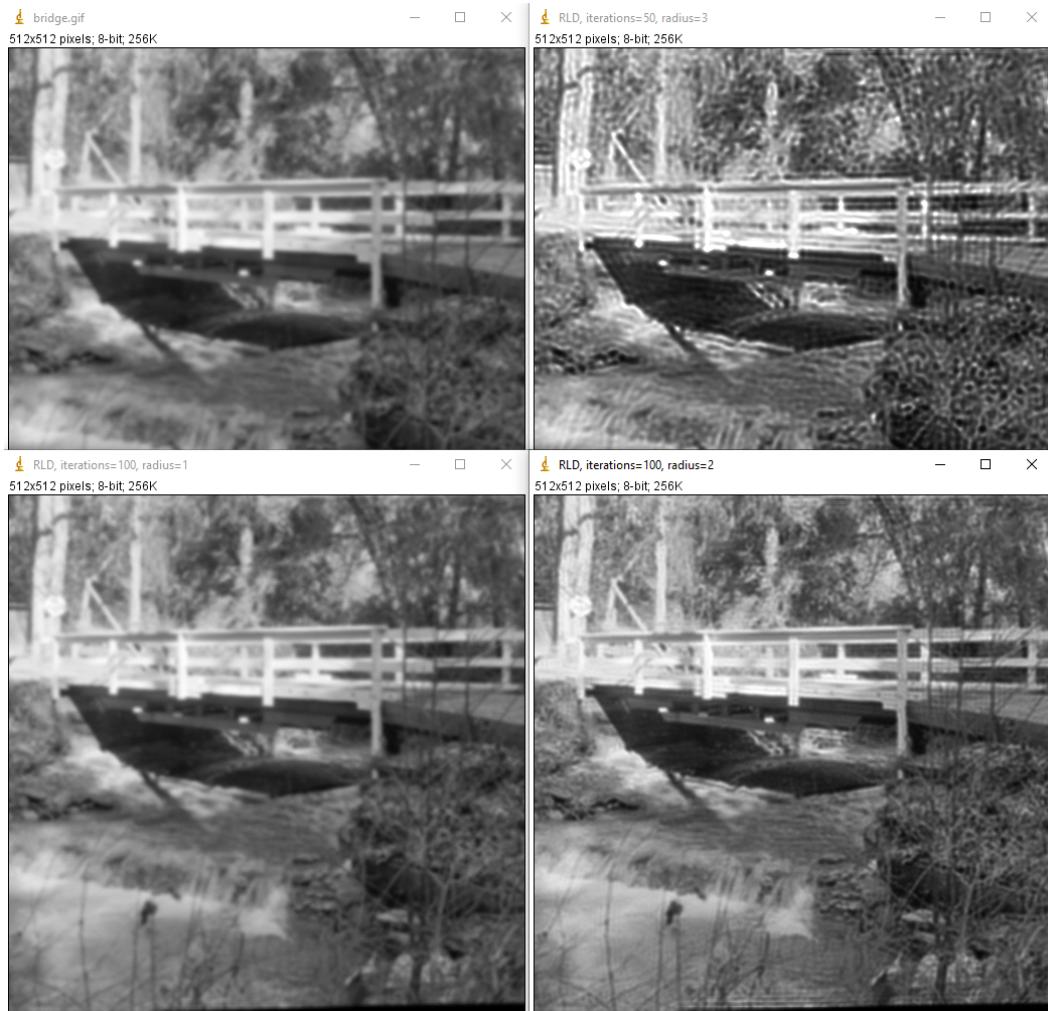


Abbildung 1.36: Links oben Original, Rechts-Oben RDL (Radius=3, Iterationen=50), Links-Unten RDL (Radius=1, Iterationen=100), Rechts-Unten RDL (Radius=2, Iterationen=100).

c) Verbesserungen im Source Code

Bei Update vom Guessbild wird noch eine Korrektur durchgeführt. Sollte es zu einem 0.0 beim Berechnen des Koeffizientenbildes kommen (bei double eher selten), dann wird hier der Divisor auf einen sehr kleinen Wert nahe Null gesetzt. Division durch Null ist damit abgefangen und liefert fast das gleiche Ergebnis (siehe Codezeile 37). Die zweite Verbesserung wurde durch das Entfernen der Artefakte implementiert, wie oben beschrieben.

Zusammenfassung und Anmerkungen

Die Übung 1.6 und 1.8 haben einen sehr hohen Zeitaufwand abverlangt (ca. 12h mit Tests und Dokumentation), weil man lange testen muss, ob die Algorithmen richtig

funktionieren. Hier würde ich es als starke Verbesserung empfinden, wenn ein vorgegebenes Bild mit den verwendeten Parametern der Übung beigefügt wird. Wenn man hier sicher sein kann, kann man viel eher mit den Testfällen beginnen.