Diffusionsgleichungen in der Bildverarbeitung

Verfasser: Alexander Klemps

1. Einleitung und Motivation

Eine klassische Aufgabe in der mathematischen Bildverarbeitung ist die Extraktion von Mustern in Bildern, wie zum Beispiel Erkennung von Kanten. In diesem Zusammenhang wurden in der Vergangenheit sehr erfolgreich lineare Kantendetektoren wie der *Canny-Edge-Detector*, der *Sobel-Detector* und verwandte Modelle angewendet. Diese Modelle sind jedoch anfällig für Bildfehler oder erkennen Strukturen, die mitunter für den weiteren Arbeitsprozess unwesentlich wenn nicht sogar hinderlich sind.

Abbildung 1: Kantenerkennung. V.l.n.r: Originalbild, bearbeitetes Bild, Kantenbild

Dementsprechend wären Methoden zur Priorisierung interessanter Strukturen bzw. Kanten und zur Minimierung von Bildfehlern äußerst hilfreich. Genau diese Aufgabe erfüllen Diffusionsfilter, insbesondere solche von nichtlinearer Struktur. Einige ausgewählte Diffusionsfilter sollen im Folgenden behandelt und auf ihre mathematischen Eigenschaften wie Wohlgestelltheit, Stabilität und Konvergenz untersucht werden. Zuletzt soll mit Implementierungen und Experimenten die Praktikabilität der Diffusionsfilter verifiziert werden.

Abbildung 2: Rauschminimierung. V.l.n.r: Originalbild, gestörtes Bild, entstörtes Bild

2. Grundlagen

2.1. Was ist ein Bild?

Um überhaupt über mathematische Methoden in der Bildverarbeitung reden zu können, muss erst einmal klar sein, wie ein Graubild mathematisch repräsentiert werden kann. Technisch gesehen sind Graubilder zweidimensionale Felder (*Arrays*) mit $N = n \cdot m$ Werten, wobei n und m für Höhe und Breite des Bildes in Pixeln stehen. Jedem Pixel wird eine Zahl zugeschrieben, welche dessen Intensität repräsentiert. Üblicherweise werden dazu Gleitkommazahlen im Intervall $[0, 1]$ oder positive ganze Zahlen im Bereich $[0, 255]$ verwendet.
Für die folgenden theoretischen Betrachtungen ist es sinnvoll, ein Graubild als Bild einer Funktion $f : \mathbb{R}^2 \to \mathbb{R}_+$ aufzufassen.

Definition. *Betrachte ein reales Bild B der Breite $n_1 \in \mathbb{N}$ und Höhe $n_2 \in \mathbb{N}$ mit Intensitätswerten $b_{ij} \in \mathcal{W} \subset \mathbb{R}_+$, wobei $i \in \{1, \dots, n_1\}$ und $j \in \{1, \dots, n_2\}$ sei. Es sei weiter $\Omega := (0, k_1) \times (0, k_2)$ mit $k_1, k_2 \in \mathbb{R}_+ \backslash \{0\}$. Definiere*

$$h_1 := \frac{k_1}{n_1}, \quad h_2 := \frac{k_2}{n_2}$$

sowie $N = n_1 \cdot n_2$ Punkte (x_i, y_i), $i \in \{1, \dots, n_1\}$, $j \in \{1, \dots, n_2\}$ durch

$$x_i := (i - \frac{1}{2})h_1, \quad y_j := (j - \frac{1}{2})h_2.$$

Sei $f : \Omega \subset \mathbb{R}^2 \to \mathcal{W}$, $f \in L^1(\Omega)$, so dass für alle $i \in \{1, \dots, n_1\}$ und $j \in \{1, \dots, n_2\}$ gilt

$$f(x_i, y_j) = b_{ij}.$$

*Dann ist Ω der **Bildbereich** des **Bildes** f.*

2.2. Isotropie und Anisotropie

Satz 2.1 (Fick'sches Gesetz). *Eine Konzentrationsänderung ∇u erzeugt einen entgegengesetzt gerichteten Strom*

$$j = -D \cdot \nabla u, \tag{1}$$

*wobei D symmetrisch und positiv definit ist. D wird als **Diffusionstensor** bezeichnet.*

Diffusion beschreibt nur den Transport von Masse, nicht deren Erzeugung oder Vernichtung. Diese Massenerhaltung wird durch die **Kontinuitätsgleichung**

$$\partial_t u = -\mathrm{div}(j). \tag{2}$$

beschrieben. Einsetzen des Fick'schen Gesetzes (1) in die Kontinuitätsgleichung (2) führt dann auf die Diffusionsgleichung

$$\partial_t u = \mathrm{div}(D \cdot \nabla u), \tag{3}$$

deren Verhalten im Folgenden für verschiedene Diffusionstensoren D untersucht werden soll. Wesentlich ist dabei die Unterscheidung von **Isotropie** und **Anisotropie**.

Definition. *Der durch Gleichung (3) beschriebene Diffusionsprozess wird als* ***isotrop*** *bezeichnet, falls ∇u und j parallel zueinander sind. Anderenfalls heißt er* ***anisotrop****.*

Bemerkung. In den folgenden Kapiteln werden Diffusionsmodelle der Form (3) für verschiedene D betrachtet.

3. Verschiedene Diffusionsmodelle

3.1. Homogene Wärmeleitungsgleichung

Definition. *Gegeben seien $f \in L^1(\Omega)$ und das AWP*

$$\begin{cases} \frac{\partial u}{\partial t}(t,\, x) = div(\nabla u) = \triangle u(t,\, x),\ x \in \Omega,\ t \geq 0 \\ u(0,\, x) = f(x). \end{cases} \tag{4}$$

Gleichung (4) wird auch als ***homogene Wärmeleitungsgleichung*** *bezeichnet.*

Bemerkung. Die Wärmeleitungsgleichung (4) ist **linear**, **parabolisch** und **isotrop**.

Von besonderer Bedeutung für die Lösung des AWP (4) ist die Funktion

$$K_\sigma(x) := \frac{1}{2\pi\sigma^2} \cdot \exp\left(-\frac{|x|^2}{2\sigma^2}\right), x \in \mathbb{R}^2,\ \sigma > 0, \tag{5}$$

wie der folgende Satz zeigt.

Satz 3.1. *Es sei u_0 eine stetige und beschränkte Fortsetzung von f auf $\mathbb{R}^2$. Dann gelten*

(1) $u(t,\, x) := (K_{\sqrt{2t}} * u_0)(x) \in C^\infty\left((0, \infty) \times \mathbb{R}^2\right).$

(2) u löst das AWP (4).

(3) u erfüllt ein Maximum- bzw. Minimumprinzip

$$\min_{x \in \mathbb{R}^2} u_0(x) \leq u(t,\, x) \leq \max_{x \in \mathbb{R}^2} u_0(x).$$

Abbildung 3: Gaussian Blur nach $t = 0,\, 13.5,\, 18\, s$

3.2. Perona-Malik-Filter

Definition. *Es sei* $g : [0, \infty) \to (0, \infty)$ *beliebig oft stetig differenzierbar und* $T > 0$. *Das Anfangs-Randwert-Problem*

$$\begin{cases} \frac{\partial u}{\partial t}(t,\, x) = div(g(|\nabla u|^2) \cdot \nabla u) \ \textit{in}\ (0,T) \times \Omega, \\ \frac{\partial u}{\partial \nu}(t,\, x) = 0 \ \textit{auf}\ (0,T) \times \partial\Omega, \\ u(0,\, x) = f(x) \ \textit{in}\ \Omega, \end{cases} \tag{6}$$

mit äußerer Normale $\nu(x)$ *an* Ω *in* $x \in \partial\Omega$ *heißt dann* ***Perona-Malik-Filter****.*

Bemerkung. Die dem Perona-Malik-Filter zugrunde liegende Diffusionsgleichung ist offenbar **nichtlinear**. Sie wird in der Literatur oft als anisotrop bezeichnet, ist aber gemäß der Definition von Isotropie und Anisotropie in Abschnitt (2.2) **isotrop**. Ob sie wie die homogene Wärmeleitungsgleichung (4) ebenfalls **parabolisch** ist, wird durch die Funktion g bestimmt.

Folgende wünschenswerte Eigenschaften beeinflussen die Wahl von g:

(a) In Regionen mit geringer Pixeldifferenz (Flächen) soll isotrop geglättet werden.

(b) An Kanten soll der Glättungsvorgang gestoppt werden.

(c) Gleichung (6) soll vom parabolischen Typ sein.

Gleichung (6) verhält sich für $g = 1$ wie die homogene Wärmeleitungsgleichung (4). Dies führt zusammen mit den Überlegungen (a) und (b) zu folgenden Forderungen an g

1. $g(s)$ soll monoton fallend sein für $s > 0$,
2. $g(0) = 1$ und $\lim\limits_{s \to +\infty} g(s) = 0$

Weitere Forderungen an g ergeben sich aus der Darstellung von Gleichung (6) bezüglich der Tangentialrichtung T und der Normalrichtung N an die Höhenlinie, auf der sich der betrachtete Pixel befindet. Diese sind für ein x mit $|\nabla u(x)| \neq 0$ definiert als

$$N(x) = \frac{\nabla u(x)}{|\nabla u(x)|}, \quad (N(x)\,|\,T(x)) = 0, \quad |T(x)| = 1.$$

Definiert man $b(s) := g(s) + 2sg'(s)$, so erhält man folgende Darstellung von (6):

$$\frac{\partial u}{\partial t} = g(|\nabla u|^2)\, u_{TT} + b(|\nabla u|^2)\, u_{NN}.$$

Somit kann (6) als gewichtete Summe zweier Diffusionsprozesse tangential und normal entlang einer Kante interpretiert werden. Im Sinne der Kantenerhaltung soll natürlich stärker in Tangentialrichtung diffundiert werden. Das bedeutet, für große Gradienten soll das asymptotische Verhalten für $s \to +\infty$ des Koeffizienten $b(s) \to 0$ dominieren. Daher ergibt sich die weitere Forderung

$$\lim_{s\to+\infty} \frac{b(s)}{g(s)} = 0 \quad \Leftrightarrow \quad \lim_{s\to+\infty} \frac{sg'(s)}{g(s)} = -\frac{1}{2}.$$

Da (6) als Diffusionsprozess modelliert wurde, ergibt sich eine weitere Bedingung an g aus der Forderung nach Parabolizität von (6). Dies führt letztlich auf

$$b(s) > 0.$$

Häufig verwendete Funktionen, die den genannten Anforderungen genügen, sind zum Beispiel

$$g(s) = \frac{1}{1 + s/\lambda^2} \quad \text{und} \quad g(s) = \exp\left(-\frac{s^2}{\lambda^2}\right),\ \lambda > 0.$$

4. Ausgewählte theoretische Resultate zu nichtlinearen Filtern

Für numerische Betrachtungen ist es sinnvoll, Diffusiionsfilter diskret bezüglich des Ortes und kontinuierlich in der Zeit aufzufassen. Dies führt auf Systeme gewöhnlicher Differentialgleichungen, welche mit geeigneten Zeitintegrationsverfahren gelöst werden können. Theoretische Betrachtungen zu semidiskreten Diffusionsmodellen wurden umfangreich von Weickert in [3], S. 75 ff, durchgeführt. Die dort behandelte Filterklasse (P_s) sowie einige ausgewählte zugehörige Resultate werden in diesem Abschnitt vorgestellt.

4.1. Semidiskrete Filterklasse nach Weickert

Definition. *Eine Matrix $A = (a_{ij}) \in \mathbb{R}^{N\times N}$ ist **irreduzibel** genau dann, wenn für alle $i, j \in J$ Indizes $k_0, \ldots, k_s \in J$ mit $k_0 = i$ und $k_s = j$ existieren, so dass $a_{k_p,k_{p+1}} \neq 0$ für jedes $p \in \{1, \ldots, s\}$ gilt.*

Definition. *Es sei $f \in \mathbb{R}^N$. Die Filterklasse (P_s) ist dann definiert als das AWP*

$$\frac{du}{dt} = A(u)u, \quad u(0) = f, \tag{7}$$

wobei $A = (a_{ij})$ die folgenden Eigenschafte besitze:

(S1) $A \in C\left(\mathbb{R}^N, \mathbb{R}^{N\times N}\right)$ sei Lipschitz-stetig auf jeder beschränkten Teilmenge von $\mathbb{R}^N$.

(S2) A sei symmetrisch, für alle $i, j \in J$ und $u \in \mathbb{R}^N$ gelte $a_{ij}(u) = a_{ji}(u)$.

(S3) Für alle $i \in J$ und $u \in \mathbb{R}^N$ gelte $\sum_{j\in J} a_{ij}(u) = 0$.

(S4) Für alle $i, j \in J$, $i \neq j$, und alle $u \in \mathbb{R}^N$ gelte $a_{ij}(u) \geq 0$.

(S5) Für alle $u \in \mathbb{R}^N$ sei A irreduzibel.

4.1.1. Wohlgestelltheit und Maximumprinzip

Satz 4.1. *Es gelten die in (P_s) definierten Voraussetzungen.*

(1) Für jedes $T > 0$ besitzt das Problem (P_s) eine eindeutige Lösung $u(t) \in C^1\left([0, T],\ \mathbb{R}^N\right)$.

(2) Die Lösung $u(t)$ erfüllt für alle $i \in J$ und $t \in [0, T]$

$$\min_{j\in J} f_j \leq u_i(t) \leq \max_{j\in J} f_j, .$$

(3) Die Lösung von (P_s) hängt stetig von f und der rechten Seite A des Systems ab.

4.1.2. Ljapunov-Funktionen und Stabilität

Im diesem Abschnitt sollen Konvergenz- und Stabilitätseigenschaften von Lösungen der Filterklasse (P_s) untersucht werden. Ein wichtiges Hilfsmittel aus der Theorie dynamischer Systeme sind dabei Ljapunov-Funktionen. Im folgenden werden solche für die Klasse (P_s) wie in [3], S. 81 ff, konstruiert, mittels welcher eine Konvergenzaussage bewiesen wird.

Lemma 4.2. *Der mittlere Grauwert*

$$\mu := \frac{1}{N} \sum_{j \in J} f_j$$

ist invariant unter Filtern der Klasse (P_s). Es gilt für alle $t > 0$

$$\frac{1}{N} \sum_{j \in J} u_j(t) = \mu.$$

Beweis. Mit der Symmetrie (S2) der Matrix A folgt aus (S3) für jedes $k \in J$

$$0 = \sum_{j \in J} a_{kj}(u) = \sum_{j \in J} a_{jk}(u).$$

Somit gilt für alle $t \geq 0$

$$\frac{d}{dt} \sum_{j \in J} u_j = \sum_{j \in J} \frac{du_j}{dt} = \sum_{j \in J} \sum_{k \in J} a_{jk}(u) u_k = \sum_{k \in J} \left(\sum_{j \in J} a_{jk}(u) \right) u_k = 0.$$

Also ist $\sum_{j \in J} u_j(t)$ konstant für alle $t \in [0, \infty)$, woraus die Behauptung folgt.

□

Satz 4.3 (Ljapunov-Funktion). *Es sei $u(t) \in C^1([0, \infty), \mathbb{R}^N)$ die Lösung des Problems (P_s). Weiter seien μ der mittlere Grauwert, $a := \min_{j \in J} f_j$, $b := \max_{j \in J} f_j$ und $c := (\mu, \mu, \ldots, \mu)^T \in \mathbb{R}^N$.*
Dann ist für jedes $r \in C^1([a, b])$ mit auf $[a, b]$ streng monoton wachsender Ableitung r' eine im Punkt $u \equiv c$ strikte Ljapunov-Funktion V für (P_s) definiert durch

$$V(t) := \Phi(u(t)) := \sum_{i \in J} r(u_i(t)). \tag{8}$$

Beweis. Wir zeigen zunächst, dass V eine nach t stetig differenzierbare Ljapunov-Funktion ist.

Da $u(t) \in C^1([0,\infty),\mathbb{R}^N)$ und $r \in C^1([a,b])$ gelten, folgt sofort $V \in C^1([0,\infty))$. Um zu zeigen, dass V eine strikte Ljapunov-Funktion ist, muss noch $\dot{V}(t) < 0$ für alle u mit $A(u(t))u(t) \neq 0$ und alle $t > 0$ nachgewiesen werden. Es gilt zunächst

$$\begin{aligned}\dot{V}(t) &= \sum_{i=1}^{N} \frac{du_i}{dt} r'(u_i) = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}(u)u_j\, r'(u_i) \\ &\overset{(S3)}{=} \sum_{i=1}^{N}\left(\sum_{j=1}^{N} a_{ij}(u)u_j\, r'(u_i) - \sum_{j=1}^{N} a_{ij}(u)u_i\, r'(u_i)\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}(u)(u_j-u_i)\, r'(u_i) \\ &= \sum_{i=1}^{N}\left(\sum_{j=i+1}^{N} + \sum_{j=1}^{i-1}\right) a_{ij}(u)(u_j-u_i)\, r'(u_i) \qquad (9)\end{aligned}$$

Durch Verwendung eines neuen Laufindex erhält man für die erste Summe über j

$$\sum_{j=i+1}^{N} a_{ij}(u)(u_j-u_i)r'(u_i) = \sum_{k=1}^{N-i} a_{i,i+k}(u)(u_{i+k}-u_i)\, r'(u_i).$$

Anstatt in der zweiten Summe über j zeilenweise Summanden mit Vorfaktor a_{ij} bis zum Index $j = i-1$ zu addieren, summiert man stattdessen spaltenweise. Dies führt auf

$$\sum_{j=1}^{i-1} a_{ij}(u)(u_j-u_i)r'(u_i) = \sum_{k=1}^{N-i} a_{i+k,i}(u)(u_i-u_{i+k})\, r'(u_{i+k})$$

Beide Umformungen in (9) eingesetzt ergeben unter Verwendung der Symmetrie (S2)

$$\begin{aligned}\dot{V}(t) &= \sum_{i=1}^{N}\sum_{k=1}^{N-i} a_{i,i+k}(u)(u_{i+k}-u_i)\, r'(u_i) + \sum_{i=1}^{N}\sum_{k=1}^{N-i} a_{i+k,i}(u)(u_i-u_{i+k})\, r'(u_{i+k}) \\ &= \sum_{i=1}^{N}\sum_{k=1}^{N-i} a_{i,i+k}(u)(u_{i+k}-u_i)\ (r'(u_i)-r'(u_{i+k})) \qquad (10)\end{aligned}$$

Wegen (S4) gilt für Elemente $a_{i,i+k}(u)$ in (10) $a_{i,i+k}(u) \leq 0$ und da r' monoton wachsend auf $[a,b]$ ist, gilt $(u_{i+k}-u_i)\,(r'(u_i)-r'(u_{i+k})) \leq 0$. Insgesamt gilt also $\dot{V}(t) \leq 0,\ t \geq 0$.

Um die Striktheit von V nachzuweisen, zeigen wir, dass $\dot{V}(t) = 0$ äquivalent zu $u \equiv c$ ist. Sei dazu $\dot{V}(t) = 0$. Dann gilt mit (10)

$$0 = \dot{V}(t) = \sum_{i=1}^{N}\sum_{k=1}^{N-i} \underbrace{a_{i,i+k}(u)(u_{i+k}-u_i)\ (r'(u_i)-r'(u_{i+k}))}_{\leq 0}$$

und wegen der Symmetrie (S2) folgt für alle $i,j \in J$

$$a_{ij}(u)(u_j-u_i)\ (r'(u_i)-r'(u_j)) = 0$$

Seien $i_0, j_0 \in J$ beliebig. Da A laut (S5) irreduzibel ist, existieren Indizes $k_0, \dots, k_s \in J$ mit $k_0 = i_0$, $k_s = j_0$ und $a_{k_p,k_{p+1}} \neq 0$ für $p \in \{1, \dots, s-1\}$. Weil zusätzlich r' streng wachsend ist, folgt daraus für jedes $p \in \{1, \dots, s\}$

$$a_{k_p,k_{p+1}}(u)(u_{k_{p+1}} - u_{k_p}) \left(r'(u_{k_p}) - r'(u_{k_{p+1}})\right) = 0 \quad \Leftrightarrow \quad u_{k_p} = u_{k_{p+1}}.$$

Damit ergibt sich insgesamt

$$u_{i_0} = u_{k_0} = \cdots = u_{k_s} = u_{j_0}.$$

Da i_0 und j_0 beliebig waren, folgt $u_i = u_j$ für alle $i, j \in J$. Lemma (4.2) liefert, dass dann $u \equiv c$ gelten muss.
Gilt umgekehrt $u \equiv c.$, so folgt direkt aus (10), dass $\dot{V}(t) = 0$ ist.

□

Eine Konvergenzaussage für Lösungen der Filterklasse (P_s) lässt sich leicht mit dem folgenden Lemma beweisen, welches als Lemma 5.5.3 in [2], S. 105 f, zu finden ist.

Lemma 4.4. *Seien V eine strikte Ljapunov-Funktion für $\dot{u} = f(u)$, $f : \mathbb{R} \to \mathbb{R}^n$ für ein $n \in \mathbb{N}$ und $\tilde{u}$ eine Lösung für $t \in [0, \infty)$. Weiter sei*

$$\omega_+(\tilde{u}) := \{y \in \mathbb{R}^n \mid \exists\, (t_k)_{k\in\mathbb{N}},\, t_k \underset{k\to\infty}{\to} \infty :\ \tilde{u}(t_k) \underset{k\to\infty}{\to} y\}$$

die ω-Grenzmenge von $\tilde{u}$. Dann gilt

$$\omega_+(\tilde{u}) \subseteq \mathcal{E} := f^{-1}[0].$$

Korollar 4.5. *Es gelten die Voraussetzungen von Satz (4.3) und V sei eine strikte Ljapunov-Funktion der Form (8) für (P_s). Dann ist $\lim\limits_{t\to\infty} u(t) = c$.*

Beweis. Es ist $V \in C^1([0, \infty))$. Daher gilt mit der Kettenregel

$$\dot{V}(t) = \frac{d\Phi(u(t))}{dt} = (\nabla\Phi \mid A(u)u).$$

Für alle $u^* \in \mathcal{E}$ gilt also $\dot{V}(t) = 0$ und somit ist $\mathcal{E} \subseteq \dot{V}^{-1}[0]$. Wie im Beweis von Satz (4.3) gezeigt, gilt $\dot{V}^{-1}[0] = \{c\}$ und es folgt insgesamt $\mathcal{E} = \{c\}$.
Sei u eine Lösung von (P_s). Diese ist beschränkt für alle $t \geq 0$, nach Bolzano-Weierstraß gilt also $w_+(u) \neq \emptyset$. Lemma (4.4) liefert nun $w_+(u) = \{c\}$, was gleichbedeutend mit $\lim\limits_{t\to\infty} u(t) = c$ ist.

□

Für spezielle Funktionen r erhält man mit Satz (4.3) weitere Informationen über interessante Funktionen. Das folgende Lemma fasst dies für $r(s) := |s|^p,\ r(s) := (s-\mu)^{2n}$ und $r(s) := s\ln(s)$ zusammen, wobei $p \geq 2$ und $n \in \mathbb{N},\ n > 0$, seien.

Korollar 4.6. *Es gelten die in (P_s) definierten Voraussetzungen. Dann sind folgende Funktionen für $t \in [0, \infty)$ monoton fallend:*

(1) $||u(t)||_p$ *für alle* $p \geq 2$.

(2) $M_{2n}[u(t)] := \frac{1}{N} \sum_{j=1}^{N} (u_j(t) - \mu)^{2n}$ *für alle* $n \in \mathbb{N}\backslash\{0\}$.

(3) $H[u(t)] := \sum_{j=1}^{N} u_j(t)\ \ln(u_j(t))$, *falls* $\min_{j \in J} f_j > 0$ *gilt.*

Bemerkung. Die in Korollar (4.6) in (2) und (3) gelisteten Funktionen erlauben eine statistische Interpretation des Verhaltens von Filtern der Klasse (P_s). Fasst man die Pixelwerte eines Bildes f als Realisierungen einer Zufallsgröße Z_f auf, so beschreibt (2) für $n = 1$ gerade die Varianz dieser Zufallsgröße. Diese ist monoton fallend, das Bild wird in Richtung des mittleren Grauwerts geglättet.
Die Funktion $H[u(t)]$ entspricht gerade dem Betrag der Entropie, welche demnach monoton wachsend ist. Sie ist ein Maß für den Informationsgehalt, welcher mit zunehmender Zeit sinkt. Dies geht einher mit der Konvergenz in Richtung eines homogenen Graubildes mit Wert μ, welches den geringsten Informationsgehalt vorweist.

4.2. Semidiskreter Perona-Malik-Filter

Im Folgenden soll eine **Ortsdiskretisierung** des Perona-Malik-Filters

$$\partial_t u = \operatorname{div}(g(|\nabla u|^2) \cdot \nabla u) = \partial_x(g(|\nabla u|^2)\partial_x u) + \partial_y(g(|\nabla u|^2)\partial_y u) \tag{11}$$

in einem inneren Pixel $(i,\, j)$ vorgenommen werden. Durch lineare Interpolation erhält man den Mittelwert

$$g_{i+\frac{1}{2},j} := \frac{1}{2}(g_{ij} + g_{i+1,j})$$

an der Stelle $(i+\frac{1}{2},\, j)$. Weitere Diskretisierungen können mit Hilfe vorwärtsgenommener und zentraler Differenzenquotienten vorgenommen werden.

$$\begin{aligned} [g\partial_x u]_{i+\frac{1}{2},j} &\approx g_{i+\frac{1}{2},j}\left(\frac{u_{i+1,j} - u_{ij}}{h_1}\right) =: \varphi_{i+\frac{1}{2},j} \\ [\partial_x(g\partial_x u)]_{ij} &\approx \left(\frac{\varphi_{i+\frac{1}{2},j} - \varphi_{i-\frac{1}{2},j}}{h_1}\right) \end{aligned}$$

Einsetzen der Definitionen von $g_{i+\frac{1}{2},j}$ und $\varphi_{i+\frac{1}{2},j}$ liefert

$$[\partial_x(g\partial_x u)]_{ij} \approx \frac{1}{2h_1^2}\left((g_{i+1,j} + g_{ij})(u_{i+1,j} - u_{ij}) - (g_{ij} + g_{i-1,j})(u_{ij} - u_{i-1,j})\right) \tag{12}$$

Eine analog zu (12) durchgeführte Diskretisierung von $\partial_y(g\partial_y u)$ führt auf folgende Diskretisierung von (11)

$$\begin{aligned} \frac{du_{ij}}{dt} &= \frac{1}{2h_1^2}\left((g_{i+1,j} + g_{ij})(u_{i+1,j} - u_{ij}) - (g_{ij} + g_{i-1,j})(u_{ij} - u_{i-1,j})\right) \\ &+ \frac{1}{2h_2^2}\left((g_{i,j+1} + g_{ij})(u_{i,j+1} - u_{ij}) - (g_{ij} + g_{i,j-1})(u_{ij} - u_{i,j-1})\right) \end{aligned}$$

Eine kompaktere Schreibweise erhält man mit

$$\frac{du_k}{dt} = \sum_{n=1}^{2} \sum_{l \in \mathcal{N}_n(k)} \frac{g_l + g_k}{2h_l^2}(u_l - u_k), \tag{13}$$

wobei $\mathcal{N}_n(k)$ die Menge der Nachbarpixel des Pixels k entlang der n-ten Koordinatenachse sei. Randpixel besitzen jeweils nur einen Nachbarpixel.
In Matrix-Vektor-Notation wird (13) zu

$$\frac{du}{dt} = A(u)u$$

mit einer Matrix $A(u) = (a_{kl}(u))_{kl}$ der Gestalt

$$a_{kl} := \begin{cases} \dfrac{g_k + g_l}{2h_n^2} & (l \in \mathcal{N}_n(k)), \\ -\sum\limits_{n=1}^{2} \sum\limits_{l \in \mathcal{N}_n(k)} \dfrac{g_l + g_k}{2h_l^2} & (k = l), \\ 0 & (\text{sonst}). \end{cases} \tag{14}$$

Um die Resultate für die Filterklasse (P_s) anwenden zu können, müssen zunächst die Eigenschaften (S1) bis (S5) für die Matrix A nachgewiesen werden.
Um die entsprechende Regularität der Matrix A zu erhalten, kann u mit einem Gauß-Kern K_σ für ein $\sigma > 0$ gefalten werden. Dann sind $u_\sigma(x) := K_\sigma * u \in C^\infty(\mathbb{R}^2, \mathbb{R})$. Da auch $g \in C^\infty(\mathbb{R})$ ist, folgt $A \in C^\infty(\mathbb{R}^N, \mathbb{R}^{N \times N})$. Damit gilt (S1).
Die Symmetrie von A folgt aus deren Konstruktion (14) und der Symmetrie der Nachbarschaftsrelation

$$l \in \mathcal{N}_n(k) \quad \Leftrightarrow \quad k \in \mathcal{N}_n(l).$$

Aus der Konstruktion von A folgt auch sofort, dass die Zeilensummen verschwinden, also (S3). (S4) folgt aus der Positivität der Funktion g.
Um zu zeigen, dass A irreduzibel ist, seien k, $l \in J$ zwei beliebige Pixel. Gilt $k = l$, so gilt $a_{kk} < 0$ und $k = k_0 = k_s = k_l$ ist eine triviale Folge von Pixeln. Für $k \neq l$ seien $k = k_0, k_1, \ldots, k_{s-1}, k_s = l \in J$ geeignete Indizes so, dass k_p und k_{p+1} Nachbarn für alle $p \in \{0, \ldots, s-1\}$ sind. Dann gilt $a_{k_p, k_{p+1}} \neq 0$ für alle $p \in \{0, \ldots, s-1\}$ und A ist damit irreduzibel.

Implementierung. Die hier vorgestellte Implementierung wurde für Präsentationszwecke und zur praktischen Überprüfung der theoretischen Resultate in *Julia* implementiert. Der Einfachheit halber wurde das explizite Euler-Verfahren als Zeitintegrationsverfahren verwendet, was aus Stabilitätsgründen nur eine maximale Zeitschrittweite von 0.25 zulässt. Die Pixelwerte lagen bei den Testbildern im Bereich $[0, 1]$, die Resultate sind auf der nachfolgenden Seite abgebildet.

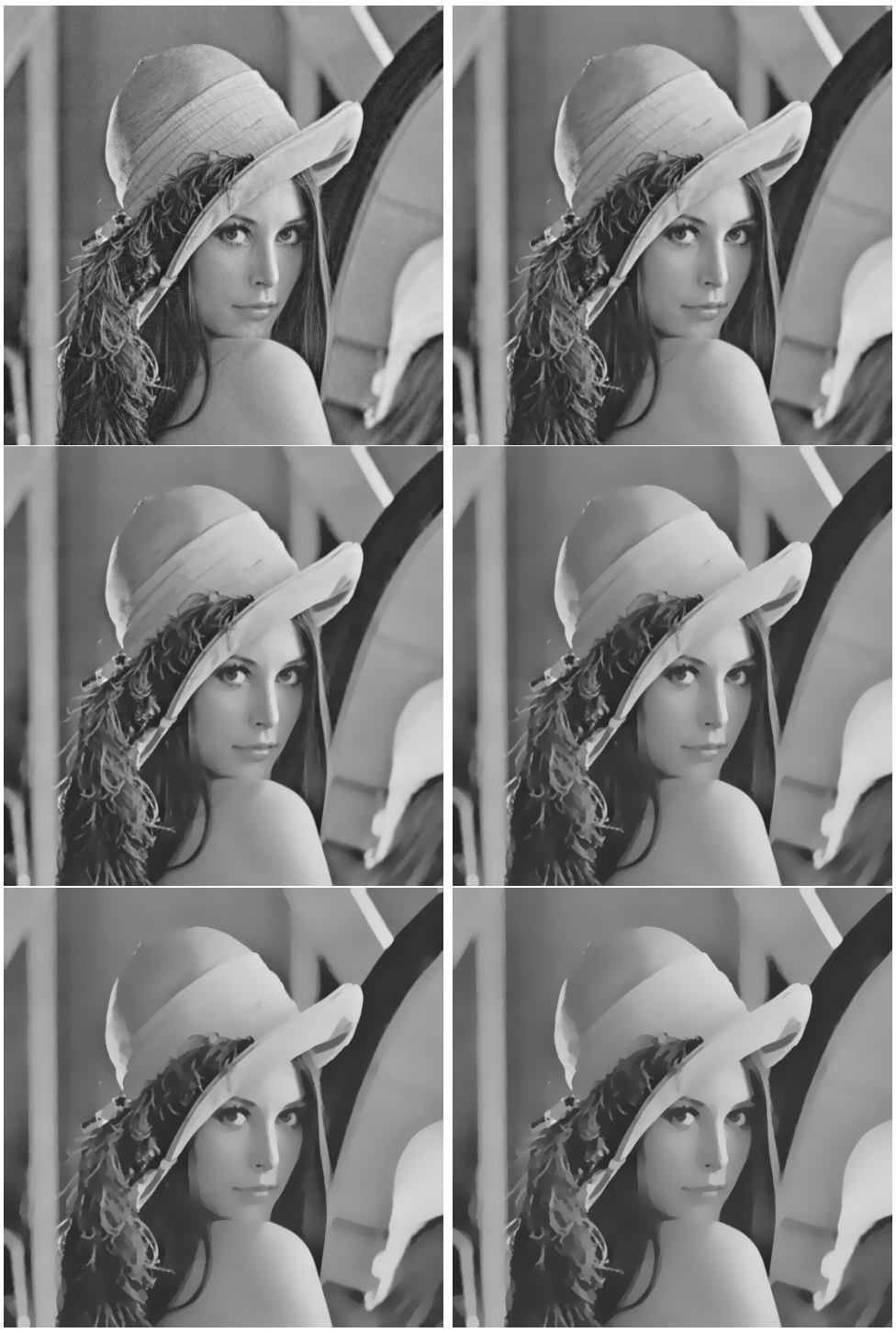

Abbildung 4: Perona-Malik-Filter ($\lambda = 8 \cdot 10^{-3}$)
nach $t = 0,\ 20,\ 50,\ 100,\ 150,\ 200\,s$

Literaturverzeichnis

[1] Gilles Aubert und Pierre Kornprobst. *Mathematical Problems in Image Processing*. 2002.

[2] Jan Prüß und Mathias Wilke. *Gewöhnliche Differentialgleichungen und Dynamische Systeme*. 2010.

[3] Joachim Weickert. *Anisotropic Diffusion in Image Processing*. 1998.

A. Implementierung des Perona-Malik-Filters

```
# Alexander Klemps, 2019

using Images, Colors, Plots, TestImages

function g(s::Array{Float64, 2}, lambda::Float64)::Array{Float64, 2}
    return 1 ./ (1 .+ s ./ lambda^2)
end;

function PeronaMalik(image::Array{Float64, 2},
                     N::Int64, tau=0.25)::Array{Float64, 2}

    h, w = size(image);
    for i = 1:N
        padded = padarray(image, Pad(1, 1));
        dN = padded[1:h, 1:w] .- padded[0:h-1, 1:w];
        dS = padded[2:h+1, 1:w] .- padded[1:h, 1:w];
        dW = padded[1:h, 1:w] .- padded[1:h, 0:w-1];
        dE = padded[1:h, 2:w+1] .- padded[1:h, 1:w];

        G = g(0.25 * ((dN .+ dS) .^ 2 + (dE .+ dW) .^2), 0.008);
        G = padarray(G, Pad(1, 1));
        gN = G[1:h, 1:w] .+ G[0:h-1, 1:w];
        gS = G[2:h+1, 1:w] .+ G[1:h, 1:w];
        gW = G[1:h, 1:w] .+ G[1:h, 0:w-1];
        gE = G[1:h, 2:w+1] .+ G[1:h, 1:w];

        dt = 0.5 .* (-gN .* dN .+ gS .* dS .- gW .* dW .+ gE .* dE);
        image = image .+ tau .* dt;
    end;
    return image
end;

path = "lena.png"
img = load(path)
mat = channelview(Gray.(img)) * 1.0

@time processed = PeronaMalik(mat, 200)
processed = Gray.(processed)
```