# Notizen zur Programmieraufgabe Seamless Cloning

Johannes Hertrich

## 1 Diskrete Ableitungen

Sei $g\colon [0, N-1] \times [0, M-1] \to [0, L-1]$ ein kontinuierliches Grauwertbild. In diesem Abschnitt nehmen wir an, dass $g$ differenzierbar ist. Dann enthält $g$ eine Kante oder Textur, wenn sich der Wert von $g$ schnell ändert. Mit anderen Worten sind Informationen über Kanten oder Texturen im Gradienten von $g$ enthalten. Deshalb möchten wir eine Entsprechung des Gradienten für diskrete Grauwertbilder finden.

**Vorwärtsdifferenzen.** Sei $v\colon [0, N-1] \to \mathbb{R}$ eine differenzierbare Funktion und $\boldsymbol{u} = (u_i)_{i=0}^{N-1}$ gegeben durch $u_i = v(i)$ das diskrete Gegenstück dazu. Dann gilt für $i = 0, ..., N-2$, dass

$$v'(i) = \lim_{h \to 0} \frac{v(i+h) - v(i)}{h}$$

Unter der Annahme, dass $N$ groß ist und sich die Ableitung $v'$ nicht stark verändert, approximieren wir jetzt den Grenzwert $h \to 0$ durch $h = 1$. Dann erhalten wir

$$v'(i) \approx v(i+1) - v(i) = u_{i+1} - u_i.$$

Das motiviert die folgende Definition.

**Definition 1.** Sei $\boldsymbol{u} = (u_i)_{i=0}^{N-1} \in \mathbb{R}^n$ ein Vektor, wobei wir formal $u_i = 0$ für $i \in \mathbb{Z} \setminus \{0, ..., N-1\}$ schreiben.

(i) Wir definieren den Vektor der *Vorwärtsdifferenzen* oder die *diskretisierte Ableitung* von $\boldsymbol{u}$ als

$$\boldsymbol{u}' = (u_i')_{i=0}^{N-1}, \qquad u_i' = u_{i+1} - u_i.$$

(ii) Analog definieren wir den Vektor der *Rückwärtsdifferenzen* von $\boldsymbol{u}$ als

$$\boldsymbol{u}^{\text{rück}} = (u_i^{\text{rück}})_{i=0}^{N-1}, \qquad u_i^{\text{rück}} = u_i - u_{i-1}.$$

(iii) Wir nennen

$$\boldsymbol{u}'' = (u_i'')_{i=0}^{N-1}, \quad u_i'' = (\boldsymbol{u}')_{i-1}' = u_{i+1} - 2u_i + u_{i-1}$$

die *diskretisierte zweite Ableitung* von $u$.

(iv) Sei $\boldsymbol{f} = (f_{i,j})_{i,j=0}^{N-1,M-1} \in \mathbb{R}^{N\times M}$ eine Matrix. Dann nennen wir $\nabla \boldsymbol{f}$ gegeben durch

$$\nabla \boldsymbol{f}_{i,j} = \begin{pmatrix} f_{i+1,j} - f_{i,j} \\ f_{i,j+1} - f_{i,j} \end{pmatrix}$$

den *diskreten Gradienten* von $f$.
Notation: $([\nabla \boldsymbol{f}]_0)_{i,j} = f_{i+1,j} - f_{i,j}$ und $([\nabla \boldsymbol{f}]_1)_{i,j} = f_{i,j+1} - f_{i,j}$

Die Funktion, die einen Vektor auf seine diskrete Ableitungen abbildet, ist linear und kann dementsprechend als Matrixmultiplikation geschrieben werden. Genauer gesagt gilt

$$\boldsymbol{u}' = D_N^v \boldsymbol{u}, \quad \boldsymbol{u}^{\text{rück}} = D_N^r \boldsymbol{u}, \quad \text{und} \quad \boldsymbol{u}'' = D_N^{(2)} \boldsymbol{u}$$

mit

$$D_N^v = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ & & & & -1 \end{pmatrix}, \quad D_N^r = \begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{pmatrix}, \tag{1}$$

und

$$D_N^{(2)} = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}. \tag{2}$$

Dementsprechend ist gild für $\boldsymbol{f} = (f_{i,j})_{i,j=0}^{N-1,M-1} \in \mathbb{R}^{N\times M}$, dass

$$[\nabla \boldsymbol{f}]_0 = D_N^v \boldsymbol{f}, \quad \text{und} \quad [\nabla \boldsymbol{f}]_1 = \boldsymbol{f}(D_M^v)^{\mathrm{T}}.$$

**Der Laplace-Operator und vektorisierte Bilder.** Bevor wir zur eigentlichen Programmieraufgabe kommen, benötigen wir noch einen weiteren (diskreten) Differentialoperator. Für eine offene Menge $\Omega \subset \mathbb{R}^2$ und eine zweimal differenzierbare Funktion $g\colon \Omega \to \mathbb{R}$, ist der Laplace-Operator als

$$\Delta g\colon \Omega \to \mathbb{R}, \qquad \Delta g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

definiert. Indem wir die beiden auftretenden zweiten Ableitungen durch die Matrix-Vektor-Multiplikation (2) diskretisieren, definieren wir für eine Matrix $\boldsymbol{f} = (f_{i,j})_{i,j=0}^{M-1,N-1} \in \mathbb{R}^{N\times M}$ den *diskreten Laplace-Operator* als

$$\Delta\boldsymbol{f} = D_N^{(2)}\boldsymbol{f} + \boldsymbol{f}D_M^{(2)} \in \mathbb{R}^{N,M}, \quad \text{d.h.} \quad \Delta\boldsymbol{f}_{i,j} = -4f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}$$

Wir möchten den Operator $\Delta$ im Folgenden ebenfalls als Matrix-Vektor-Multiplikation ausdrücken. Dafür benötigen wir folgende Definition.

**Definition 2.** Für zwei Matrizen $A \in \mathbb{R}^{M\times M}$ und $B \in \mathbb{R}^{N\times N}$ mit Einträgen

$$A = \begin{pmatrix} a_{0,0} & \cdots & a_{0,N-1} \\ \vdots & & \vdots \\ a_{N-1,0} & \cdots & a_{N-1,N-1} \end{pmatrix} \quad \text{und} \quad B = \begin{pmatrix} b_{0,0} & \cdots & b_{0,M-1} \\ \vdots & & \vdots \\ b_{M-1,0} & \cdots & b_{M-1,M-1} \end{pmatrix}$$

definieren wir das *Kronecker-Produkt* als Blockmatrix

$$A \otimes B \coloneqq \begin{pmatrix} a_{0,0}B & \cdots & a_{0,N-1}B \\ \vdots & & \vdots \\ a_{N-1,0}B & \cdots & a_{N-1,N-1}B \end{pmatrix} \in \mathbb{R}^{NM\times NM}.$$

Des Weiteren definieren wir die *Vektorisierung* einer Matrix $A = (a_{i,j})_{i,j=0}^{N-1,M-1} \in \mathbb{R}^{N\times M}$ als den Vektor $\mathrm{vec}(A) = (\mathrm{vec}(A)_i)_{i=0}^{MN-1} \in \mathbb{R}^{MN}$ als die Andernanderreihung der Spalten von $A$, d.h., $\mathrm{vec}(A)_{i+Nj} = a_{i,j}$. Mit diesen Definitionen erhalten wir das folgende Lemma.

**Lemma 3.** *(i) Seien $A \in \mathbb{R}^{N\times N}$, $\boldsymbol{f} \in \mathbb{R}^{N\times M}$ und $B \in \mathbb{R}^{M\times M}$. Dann gilt*

$$\mathrm{vec}(A\boldsymbol{f}B) = (B^{\mathrm{T}} \otimes A)\mathrm{vec}(\boldsymbol{f})$$

*(ii) Für $\boldsymbol{f} \in \mathbb{R}^{N\times M}$ gilt*

$$\mathrm{vec}(\Delta\boldsymbol{f}) = (I_M \otimes D_N^{(2)} + D_M^{(2)} \otimes I_N)\mathrm{vec}(\boldsymbol{f}).$$

*Beweis.* (i) Seien $b_i = \begin{pmatrix} b_{0,i} \\ \vdots \\ b_{M-1,i} \end{pmatrix}$ und $\boldsymbol{f}_i = \begin{pmatrix} f_{0,i} \\ \vdots \\ f_{N-1,i} \end{pmatrix}$ die Spalten von $B$ und $\boldsymbol{f}$.

Dann gilt

$$\mathrm{vec}(A\boldsymbol{f}B) = \begin{pmatrix} A\boldsymbol{f}b_0 \\ \vdots \\ A\boldsymbol{f}b_{M-1} \end{pmatrix} = \begin{pmatrix} A(b_{0,0}\boldsymbol{f}_0 + \cdots + b_{M-1,0}\boldsymbol{f}_{M-1}) \\ \vdots \\ A(b_{0,M-1}\boldsymbol{f}_0 + \cdots + b_{M-1,M-1}\boldsymbol{f}_{M-1}) \end{pmatrix}$$

$$= \begin{pmatrix} Ab_{0,0} & \cdots & Ab_{M-1,0} \\ \vdots & & \vdots \\ Ab_{0,M-1} & \cdots & Ab_{M-1,M-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_0 \\ \vdots \\ \boldsymbol{f}_{M-1} \end{pmatrix} = (B^{\mathrm{T}} \otimes A)\mathrm{vec}(\boldsymbol{f}).$$

Figure 1: Links: originale Bildausschnitte, rechts: Aneinandergefügte Bildausschnitte ohne unnatürliche Kanten

(ii) Dieser Teil ergibt sich aus Teil (i) mit zusammen mit der Definition

$$\begin{aligned}\mathrm{vec}(\Delta \boldsymbol{f}) &= \mathrm{vec}(D_N^{(2)} \boldsymbol{f}) + \mathrm{vec}(\boldsymbol{f} D_M^{(2)}) = \mathrm{vec}(D_N^{(2)} \boldsymbol{f} I_M) + \mathrm{vec}(I_N \boldsymbol{f} D_M^{(2)}) \\ &= (I_M \otimes D_N^{(2)} + D_M^{(2)} \otimes I_N)\mathrm{vec}(\boldsymbol{f}).\end{aligned}$$

□

**Bemerkung 4** (Umsetzung in Python)**.** Der vektorisierte Laplace-Operator ist sehr dünn besetzt. Das heißt, fast alle Einträge von $(I_M \otimes D_N^{(2)} + D_M^{(2)} \otimes I_N)$ sind 0. Solche Matrizen können in Python als „sparse" Matrizen dargestellt werden. Hier wird nicht für jeden Index $i, j$ ein Eintrag der Matrix gespeichert, sondern eine Liste von Nicht-Null-Einträgen. Bei dünn besetzten Matrizen ist das deutlich speicher- und recheneffizienter. Um solche sparsen Matrizen zu erzeugen, können in Python die Funktionen

- `scipy.sparse.eye` (sparse Diagonalmatrizen)
- `scipy.sparse.kron` (Kronecker-Produkt für sparse Matrizen)

verwendet werden.

## 2 Die Poisson-Gleichung für Seamless Cloning

**Problemstellung.** Im Folgenden möchten wir Ausschnitte aus einem Bild in ein anderes einfügen ohne unnatürliche Kanten zu erzeugen, siehe Figure 1. Dieser Prozess wird „seamless cloning", also „nahtloses Klonen" genannt. Dafür betrachten wir zunächst Grauwertbilder und verwenden folgende Notationen. Sei $f^*\colon S \to \mathbb{R}$ das Hintergrundbild mit Definitionsbereich $S \subseteq \mathbb{R}^2$. Außerdem sei $\Omega \subseteq S$ eine abgeschlossene Teilmenge von $S$ und $g\colon \Omega \to \mathbb{R}$ der einzufügende Bildausschnitt. Dabei schreiben wir $\partial\Omega$ für den Rand und $\mathring{\Omega}$ für das innere von $\Omega$. Unser Ziel ist es nun $g$ nahtlos in $f^*$ einzufügen.

Wir haben bereits bemerkt, dass die Informationen über Texturen und Kanten eines Bildes in dessen Gradienten enthalten sind. Deshalb möchten wir jetzt ein Bild $h^*\colon S \to \mathbb{R}$ berechnen, sodass

- der Hintergrund erhalten bleibt, das heißt $h^*|_{S\setminus\Omega} = f^*$,
- der ersetzte Bildausschnitt $h^*|_\Omega$ auf dem Rand von $\Omega$ mit $f^*$ übereinstimmt, das heißt $h^*|_{\partial\Omega} = f^*|_{\partial\Omega}$,
- die Gradienten von $h^*$ und $g$ auf $\Omega$ möglichst ähnlich sind, das heißt

$$\frac{1}{2}\int_\Omega \|\nabla h^* - \nabla g\|^2 \mathrm{d}(x,y)$$

ist möglichst klein.

Da $h^*$ und $f^*$ außerhalb von $\Omega$ gleich sein sollen, genügt es also folgendes Optimierungsproblem zu lösen:

$$h^*|_\Omega \in \underset{h\colon \Omega\to\mathbb{R}}{\arg\min} \frac{1}{2}\int_\Omega \|\nabla h - \nabla g\|^2 \mathrm{d}(x,y) \quad \text{subject to} \quad h|_{\partial\Omega} = f^*|_{\partial\Omega}. \tag{3}$$

**Diskretisierung.** Um dieses Optimierungsproblem numerisch zu lösen müssen wir es diskretisieren und insbesondere von kontinuierlichen zu diskreten Bildern übergehen. Der Einfachheit nehmen wir an, dass $\Omega$ rechteckig ist und diskretisieren es gleichmäßig in beide Richtungen. Dann erhalten wir das rechteckige Gitter

$$\Omega = \begin{pmatrix} \times & \times & \times & \cdots & \times & \times & \times \\ \times & \bullet & \bullet & \cdots & \bullet & \bullet & \times \\ \times & \bullet & \bullet & \cdots & \bullet & \bullet & \times \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \times & \bullet & \bullet & \cdots & \bullet & \bullet & \times \\ \times & \bullet & \bullet & \cdots & \bullet & \bullet & \times \\ \times & \times & \times & \cdots & \times & \times & \times \end{pmatrix},$$

wobei $\times$ die Elemente in $\partial\Omega$ markiert. Dementsprechend können wir $g$ und $h$ als $N \times M$ Matrizen $\boldsymbol{g} = (g_{i,j})_{i,j=0}^{N-1,M-1}$ und $\boldsymbol{h} = (h_{i,j})_{i,j=0}^{N-1,M-1}$ darstellen. Dabei ist dann

$$\begin{aligned} \Omega &= \{(i,j) : i = 0,...,N-1,\ j = 0,...,M-1\}, \\ \mathring{\Omega} &= \{(i,j) : i = 1,...,N-2,\ j = 1,...,M-2\} \quad \text{und} \\ \partial\Omega &= \{(i,j) : i \in \{0,N-1\}, j = 1,...,M-1\} \cup \{(i,j) : i = 0,...,N-1, j \in \{0,M-1\}\}. \end{aligned}$$

Indem wir die Ableitungen in (3) diskretisieren, erhalten wir dann folgendes Optimierungsproblem

$$h^*|_\Omega \in \underset{\boldsymbol{h}\in\mathbb{R}^{N\times M}}{\arg\min}\, F(\boldsymbol{h}) \coloneqq \frac{1}{2}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} \|\nabla \boldsymbol{h}_{i,j} - \nabla \boldsymbol{g}_{i,j}\|^2 \quad \text{subject to} \quad \boldsymbol{h}|_{\partial\Omega} = \boldsymbol{f}^*|_{\partial\Omega} \tag{4}$$

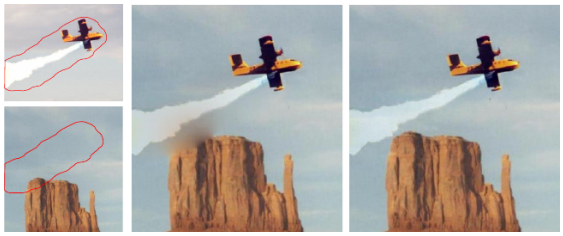

Figure 2: In manchen Fällen, wollen wir die Strukturen aus dem Hintergurndbild $\boldsymbol{f}^*$ auf $\Omega$ nicht löschen. Links: Ausgangsbilder, Mitte: keine Strukturen von $\boldsymbol{f}^*$ auf $\Omega$, Rechts: Strukturen aus Hintergrundbild $\boldsymbol{f}^*$ und $\boldsymbol{g}$ auf $\Omega$.

Um dieses Problem zu lösen, suchen wir Nullstellen von der Ableitung von $F$ bezüglich $h|_{\mathring{\Omega}}$. Dabei erhalten wir

$$\begin{aligned}\frac{\partial F}{\partial h_{i,j}} &= 4h_{i,j} - h_{i+1,j} - hi-1,j - h_{i,j+1} - h_{i,j-1} \\ &\quad - 4g_{i,j} - g_{i+1,j} - gi-1,j - g_{i,j+1} - g_{i,j-1} = -\Delta \boldsymbol{h}_{i,j} + \Delta \boldsymbol{g}_{i,j}.\end{aligned}$$

Wenn wir also die Ableitung auf 0 setzen erhalten wir das Gleichungssystem

$$\Delta \boldsymbol{h} = \Delta \boldsymbol{g} \text{ auf } \mathring{\Omega} \quad \text{mit} \quad \boldsymbol{h}|_{\partial\Omega} = \boldsymbol{f}^*|_{\partial\Omega}$$

Dies ist eine diskrete Form der so-genannten Poisson-Gleichung. Durch Lemma 3 (ii) können wir sie wie folgt vektorisieren:

$$(I_M \otimes D_N^{(2)} + D_M^{(2)} \otimes I_N)\mathrm{vec}(\boldsymbol{h}) = (I_M \otimes D_N^{(2)} + D_M^{(2)} \otimes I_N)\mathrm{vec}(\boldsymbol{g}) \text{ auf } \mathring{\Omega} \quad \text{mit} \quad \boldsymbol{h}|_{\partial\Omega} = \boldsymbol{f}^*|_{\partial\Omega}$$

Dieses Gleichungssystem kann jetzt beispielsweise mit Hilfe des CG-Verfahrens gelöst werden.

**Bemerkung 5.** Für das CG-Verfahren für sparse Matrizen kann in Python die Funktion

`scipy.sparse.linalg.cg`

verwendet werden. Achtet darauf, dass das Abbruchkriterium so gewählt ist, dass genügend Iterationen durchlaufen werden.

**Allgemeinere Vektorfelder in der Poisson-Gleichung.** Bei der bisherigen Methode werden alles Strukturen des Hintergrundbildes $\boldsymbol{f}^*$ auf $\Omega$ gelöscht. In manchen Fällen kann das zu unerwünschten Ergebnissen führen. Ein Beispiel ist in Figure 2 gegeben. Hierfür ersetzen wir den Gradienten von $g$ in (3) durch ein allgemeineres Vektorfeld $v$.

$$h^*|_\Omega \in \underset{h\colon \Omega \to \mathbb{R}}{\arg\min} \frac{1}{2} \int_\Omega \|\nabla h - v\|^2 \mathrm{d}(x,y) \quad \text{subject to} \quad h|_{\partial\Omega} = f^*|_{\partial\Omega}.$$

Erneutes Diskretisieren liefert mit $\boldsymbol{v} = (v_{i,j,k})_{i,j,k=0}^{N-1,M-1,1}$ analog zu (4) das Problem

$$h^*|_\Omega \in \underset{\boldsymbol{h}\in\mathbb{R}^{N\times M}}{\arg\min}\, F_{\boldsymbol{v}}(\boldsymbol{h}) \coloneqq \frac{1}{2}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} \|\nabla \boldsymbol{h}_{i,j} - v_{i,j}\|^2 \quad \text{subject to} \quad \boldsymbol{h}|_{\partial\Omega} = \boldsymbol{f}^*|_{\partial\Omega}.$$

Die Zielfunktion $F_{\boldsymbol{v}}$ hat hierbei die Ableitung

$$\begin{aligned}\frac{\partial F}{\partial h_{i,j}} &= 4h_{i,j} - h_{i+1,j} - hi - 1, j - h_{i,j+1} - h_{i,j-1} \\ &\quad + \underbrace{v_{i,j,0} - v_{i-1,j,0} + v_{i,j,1} - v_{i,j-1,1}}_{=:\operatorname{div} \boldsymbol{v}_{i,j}} = -\Delta \boldsymbol{h}_{i,j} + \operatorname{div} \boldsymbol{v}_{i,j},\end{aligned}$$

wobei $\operatorname{div} \boldsymbol{v} = (\operatorname{div} \boldsymbol{v}_{i,j})_{i,j=0}^{N-1,M-1}$ die diskretisierte Divergenz von $\boldsymbol{v}$ ist.. Damit erhalten wir eine allgemeinere Form der diskreten Poisson-Gleichung, nämlich

$$\Delta \boldsymbol{h} = \operatorname{div} \boldsymbol{v} \text{ auf } \mathring{\Omega} \quad \text{mit} \quad \boldsymbol{h}|_{\partial\Omega} = \boldsymbol{f}^*|_{\partial\Omega}$$

Um diese Gleichung wieder in Matrix-Vektorform zu überführen, bemerken wir, dass

$$\operatorname{div} \boldsymbol{v} = D_N^r [\boldsymbol{v}]_0 + [\boldsymbol{v}]_1 (D_M^r)^{\mathrm{T}}, \qquad [\boldsymbol{v}]_k = (v_{i,j,k})_{i,j=0}^{N-1,M-1},$$

wobei $D_N^r$ die Matrix der Rückwärtsdifferenzen aus (1) ist.

Abschließend müssen wir noch das Vektorfeld $\boldsymbol{v}$ passend wählen, um Objekte aus beiden Bildern zu überlagern. Dafür übernehmen wir an jeder Stelle $(i, j)$ immer den Gradienten mit größerer Norm, das heißt wir wählen

$$\boldsymbol{v}_{i,j} = \begin{cases} \nabla \boldsymbol{f}_{i,j}^*, & \text{falls } \|\nabla \boldsymbol{f}_{i,j}^*\| > \|\nabla \boldsymbol{g}\|, \\ \nabla \boldsymbol{g}_{i,j}, & \text{sonst.} \end{cases}$$

**Bemerkung 6** (Farbbilder)**.** Für Seamless Cloning für Farbbilder lösen wir das Problem für jeden Farbkanal separat.