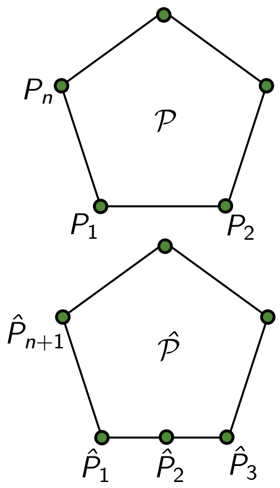
Erweiterung auf nicht-konvexe Gebiete

- Lemma (ohne Beweis):
  Sei ein konvexes Polygon $\mathcal{P}$ gegeben.
  Bezeichne die baryzentr. Koord. eines Punktes X bzgl. $\mathcal{P}$ mit w_i, i=1…n.
  $\mathcal{P}$ werde nun durch Einfügen eines Punktes verfeinert. Bezeichne dieses verfeinerte Polygon mit $\hat{\mathcal{P}}$.
  Bezeichne die baryzentr. Koord. von X bzgl. $\hat{\mathcal{P}}$ mit $\hat{w}_i$, i=1…n+1.
  Es gilt:

$$\sum_{i=1}^{n+1} \hat{w}_i = \sum_{i=1}^{n} w_i$$

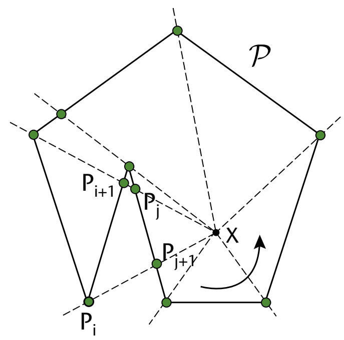


- Konsequenz: damit sind auch die $\lambda' s$ für $\hat{\mathcal{P}}$ wohl definiert

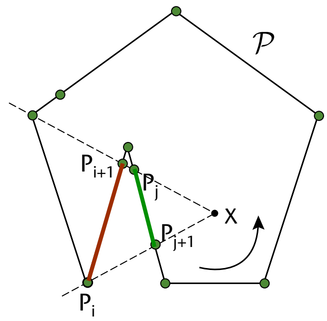
- Satz:
  Sei $\mathcal{P}$ ein beliebiges, einfaches Polygon.
  Für alle X, die nicht auf dem Rand des Polygons $\mathcal{P}$ liegen, ist

$$\sum w_i(X) \neq 0$$

- Beweis:
  - Annahme: X im Inneren von $\mathcal{P}$
  - Zeichne Strahlen von X durch die Ecken von $\mathcal{P}$ $\rightarrow$ Verfeinerung von $\mathcal{P}$
  - Nenne die Verfeinerung wieder $\mathcal{P}$, und dessen Ecken $P_1,\ldots,P_n$.



- Klassifiziere Kanten in "Entry-Kante" (rot) oder "Exit-Kante" (grün)
  - Entweder gemäß Umlaufsinn; oder gemäß Strahl von X aus
- Beobachtung: Zu jeder Entry-Kante gibt es eine (näher gelegene) Exit-Kante



- Definiere für jede Kante P_iP_{i+1} den Wert

$$k_i = \left(\frac{1}{r_i} + \frac{1}{r_i + 1}\right) \tan \frac{\alpha_i}{2}$$

wobei die Winkel α_i mit Vorzeichen gemäß Umlaufsinn behaftet sind.

- Man sieht sofort: $\sum k_i = \frac{1}{2} \sum w_i$

  (Die Summanden sind nur etwas anders zusammengefasst, und es fehlt der Faktor 1/2 bei den r_i .)
- Klar ist: falls die Kante P_iP_{i+1}
  - Exit-Kante $\rightarrow k_i > 0$
  - Entry-Kante $\rightarrow k_i < 0$
- Sei P_iP_{i+1} eine Entry-Kante
- Dann existiert dazu eine Exit-Kante P_jP_{j+1} , die näher an X liegt
- Für deren Winkel gilt $\alpha_i = -\alpha_j$
- Für die Abstände gilt:

$$r_j \leq r_{i+1} \ \wedge \ r_{j+1} < r_i \quad \text{oder} \quad r_j < r_{i+1} \ \wedge \ r_{j+1} \leq r_i$$

- Damit gilt

$$k_j = \left(\frac{1}{r_j} + \frac{1}{r_{j+1}}\right) \tan\frac{\alpha_j}{2} \quad > \quad \left(\frac{1}{r_i} + \frac{1}{r_{i+1}}\right) \tan\frac{-\alpha_i}{2} = -k_i$$

- D.h.: zu jedem k_i einer Entry-Kante gibt es ein k_j einer Exit-Kante, so dass $k_i + k_j > 0$
- Also ist $\sum k_i > 0$

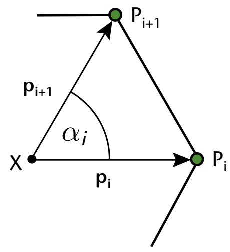
  und damit auch $\sum w_i > 0$

  für alle X im Inneren von $\mathcal{P}$

- Auch für nicht-konvexe Polygone kann man weiterhin zeigen, daß die *mean value coordinates* die Eigenschaft haben, daß:
  - λ_i auch für X auf dem Rand des Polygons wohl-definiert sind;
  - $\lambda_i(P_j) = \delta_{ij}$;
  - $\lambda_i \in \mathcal{C}^\infty$, außer an den P_j ; dort sind sie nur $\mathcal{C}^0$

Implementierung

- Praktische Berechnung des $\tan\left(\frac{\alpha_i}{2}\right)$:



$$\tan\frac{\alpha_i}{2} = \frac{1-\cos\alpha_i}{\sin\alpha_i}$$

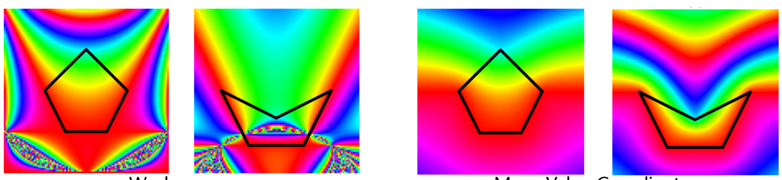
$$\cos\alpha_i = \frac{\mathbf{p}_i \cdot \mathbf{p}_{i+1}}{|\mathbf{p}_i| \cdot |\mathbf{p}_{i+1}|} \qquad \sin\alpha_i = \frac{|\mathbf{p}_i \times \mathbf{p}_{i+1}|}{|\mathbf{p}_i| \cdot |\mathbf{p}_{i+1}|}$$

Also: $$\tan\frac{\alpha_i}{2} = \frac{|\mathbf{p}_i| \cdot |\mathbf{p}_{i+1}| - \mathbf{p}_i \cdot \mathbf{p}_{i+1}}{|\mathbf{p}_i \times \mathbf{p}_{i+1}|}$$

- Falls $|\mathbf{p}_i \times \mathbf{p}_{i+1}| = 0$, dann liegt X auf der Kante;
  $\rightarrow$ Spezialbehandlung:
  1. $X = P_i$ oder $X = P_{i+1}$
  2. Sonst: linear zwischen P_i und P_{i+1} interpolieren

Anwendung: Interpolation von Farben

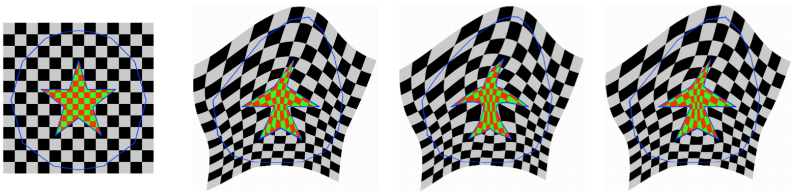
- Gegeben:
  - Ein einfaches Polygon (nicht notwendigerweise konvex)
  - An jeder Ecke eine Farbe
- Aufgabe: das Innere des Polygons mit "schönen" Farbverläufen einfärben (häufige Aufgabe z.B. in Zeichen-Software)
- Lösung:
  - Berechne für jedes Pixel im Inneren dessen baryzentrische Koordinaten bzgl. des gegebenen Polygons
  - Interpoliere die Farben der Ecken mittels dieser baryzentrischen Koord.



Wachspress Mean Value Coordinates

Anwendung: Image Warping

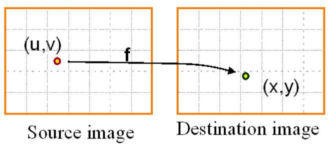
- Aufgabe: Bild gegeben, verzerre dieses durch Verschieben einiger "Kontrollpunkte"
- Beispiele:



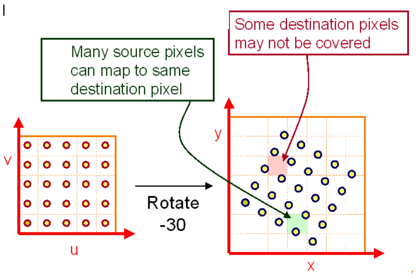
Algorithmus

- Erste Idee: *"Forward Mapping"*

```
for u = 0 .. umax:
  for v = 0 .. vmax
    x, y = f(u,v)
    dst(x,y) ← src(u,v)
```

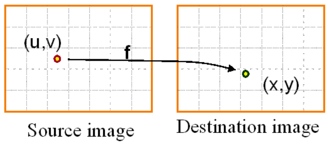


- Konstruktion von f:
  - verwende baryzentrische Koord
  - Bestimme baryz. Koord bzgl. Kontrollpunkte im Quellbild
  - Interpoliere Positionen der Kontrollpunkte im Zielbild
- Probleme:

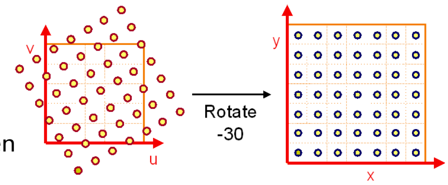


- Bessere Idee: "*Reverse Mapping*"

```
for x = 0 .. xmax:
  for y = 0 .. ymax
    u, v = f⁻¹(x,y)
    dst(x,y) ← src(u,v)
```

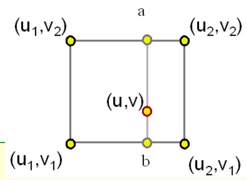


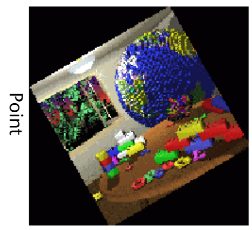
- Wieder baryzentrische Interpolation für f^{-1}, diesmal mit vertauschten Rollen
- Kleines Problem:
  - (u,v) sind keine Pixel-Koord, sondern liegen "dazwischen"
  - Man muß "*Resampling*" machen

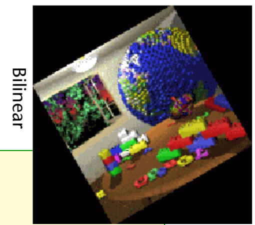


Resampling

- Einfachste Lösung: Runden
  - Ergibt schwere Artkfakte ("Aliasing"; dazu später mehr)
- Zweit-einfachste Lösung: bi-lineare Interpolation



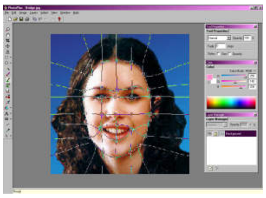


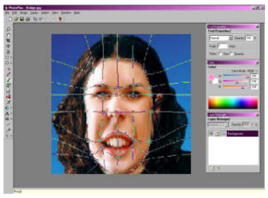



```
for x = 0 .. xmax:
  for y = 0 .. ymax
    u, v = f⁻¹(x,y)
    a = lin.interp. zwischen src(u₁,v₂) und src(u₂,v₂)
    b = lin.interp. zwischen src(u₁,v₁) und src(u₂,v₁)
    c = lin.interp. zwischen a und b
    dst(x,y) ← c
```

- Weitere Beispiele:

- Fertig integriert in Software:





Anwendung: Morphing [2004]

- Gegeben: zwei Dreiecks-Meshes M_1 und M_2 mit ...
  - genau gleich vielen Vertices und gleich vielen Dreiecken; und
  - einer Korrespondenz $\Phi : V_1 \longrightarrow V_2$ so, daß

$$P, Q, R \text{ ist ein Dreieck in } M_1 \Leftrightarrow$$

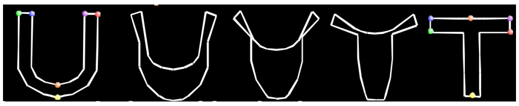
$$\Phi(P), \Phi(Q), \Phi(R) \text{ ist ein Dreieck in } M_2$$

- Aufgabe: eine gleichmäßige "Verformung" von Mesh M_1 in M_2
  - Wegen der Korrespondenz genügt es, die Koordinaten der Vertices von V_1 gleichmäßig (z.B. über 1000 Zeitschritte hinweg) so zu verändern, daß am Ende V_2 entsteht
- Terminologie: M_1 und M_2 heißen auch *"morph targets"*, oder *"source"* und *"target"*

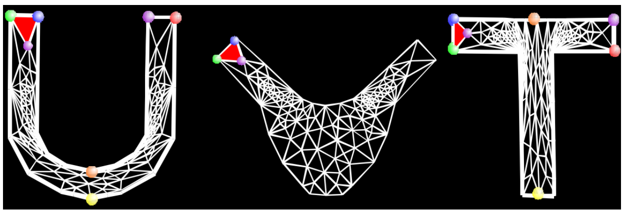
- Sei *t* der "Morph-Parameter"
- Naïve Lösung: lineare Interpolation

$$P(t) = (1-t)P + t\Phi(P)$$

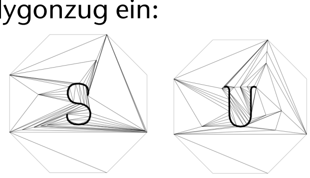
  - Beispiel:



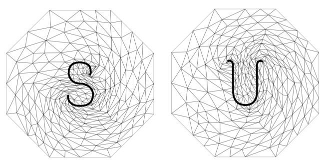
- Annahme: beide Meshes M_1 und M_2 befinden sich in der Ebene



- Schließe beide Morph-Targets in einen gemeinsamen, festen(!) Polygonzug ein:

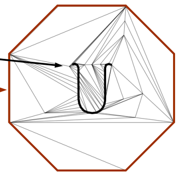


Mit wenig oder gar keinen zusätzlichen Punkten



Vielen zusätzlichen (Steiner) Punkten

- Bezeichnungen:
  - Innere Vertices $V_I = \{P_i \mid i = 1 \dots n\}$
  - Rand-Punkte $V_B = \{P_i \mid i = n+1 \dots n+k\}$
  - $N = n + k$
  - E = Menge der Kanten



- Stelle mittels verallgemeinerter baryzentr. Koordinaten ein LGS für alle Vertices auf (jeweils für M_1 und M_2):
  - Für jedes $P_i \in V_I \ , \ i = 1 \dots n$

    bestimme $\lambda_{ij} > 0 \ \forall (i,j) \in E$

    und setze $\lambda_{ij} = 0 \ \forall (i,j) \notin E$

- Damit ist

$$P_i = \sum_{i=1}^{N} \lambda_{ij} P_j \ , \quad i = 1 \dots n$$

- Etwas anders aufgeschrieben:

$$P_i - \sum_{i=1}^{n} \lambda_{ij} P_j = \sum_{i=n+1}^{n+k} \lambda_{ij} P_j \ , \quad i = 1 \dots n$$

- Mit $P_i = (x_i, y_i, z_i)$ ergeben sich also 3 LGSe:

$$\underbrace{\begin{pmatrix} 1 & -\lambda_{12} & \cdots & -\lambda_{1n} \\ -\lambda_{21} & 1 & \cdots & \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} \lambda_{1,n+1} x_{n+1} + \cdots + \lambda_{1,n+k} x_{n+k} \\ \vdots \\ \vdots \end{pmatrix}}_{\mathbf{b}}$$

  - Analog für y und z

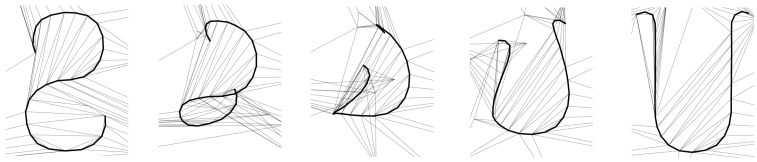
- Die (simple) Idee:
  1. Interpoliere die λ's:
  $$\lambda_{ij}^{(t)} = (1-t)\lambda_{ij}^{(1)} + t\lambda_{ij}^{(2)}$$
  2. Löse für jedes *t* die 3 LGSe
- Etwas weniger simple Idee ("intrinsisches Morphing"):
  1. Interpoliere die α's und *r*'s:
  $$\alpha_{ij}^{(t)} = (1-t)\alpha_{ij}^{(1)} + t\alpha_{ij}^{(2)} \qquad r_{ij}^{(t)} = (1-t)r_{ij}^{(1)} + tr_{ij}^{(2)}$$
  2. Berechne daraus λ(*t*)'s
  3. Löse die 3 LGSe
- Übungsaufgabe: wieviele Parameter werden in den 3 Varianten interpoliert? (für ein bestimmtes *t*)

Zur Implementierung

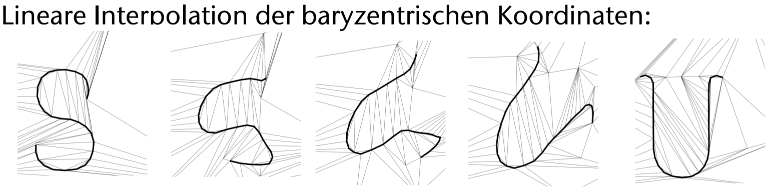
- Bemerkung:
  - Die Matrix A ist nicht notw. symmetrisch
  - Sie ist dünn besetzt
  - Sie ist größtenteils diagonal dominiert, aber keine Bandmatrix
- Verwende einen iterativen Solver
- Starte mit der Matrix von *t*-1

Resultate

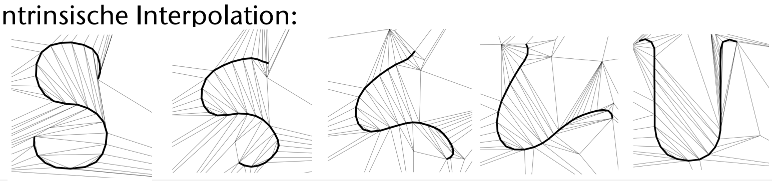
- Lineare Interpolation der Vertices:



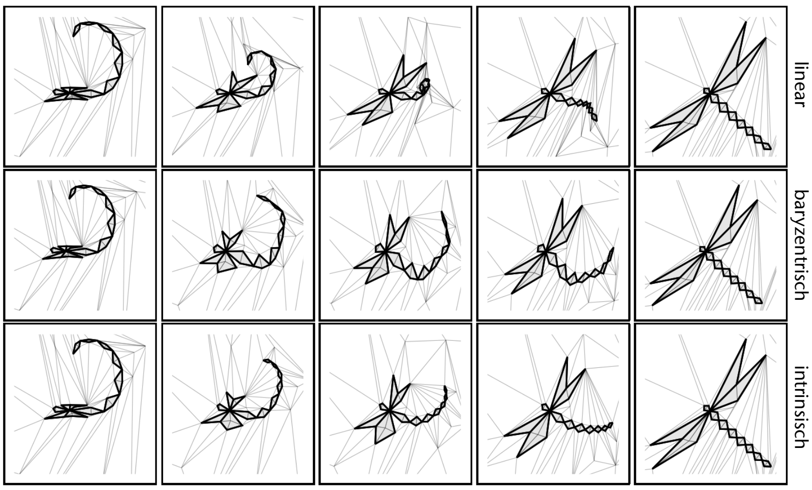
- Lineare Interpolation der baryzentrischen Koordinaten:



- Intrinsische Interpolation:



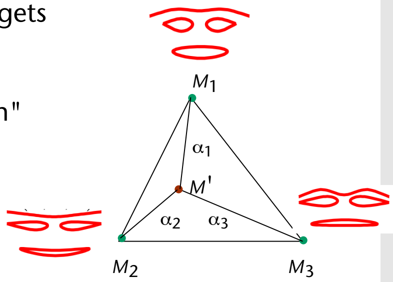
- Weiteres Beispiel:



Erweiterung

- Simultanes Morphing mehrerer Targets
- Gegeben n Morph-Targets M_i
- Aufgabe: bestimme ein "in-between"

$$M' = \sum \alpha_k M_k$$



- Idee:
  1. Bestimme die baryzentrischen Koord. $\lambda_{ij}^{(k)}$ aller M_k bzgl. eines festen Kontrollpolygons (oder Kontrollpolyeders im 3D)
  2. Interpoliere die λ's:

$$\lambda'_{ij} = \sum \alpha_k \lambda_{ij}^{(k)}$$

  3. LGSe lösen