Superpixel

30. Januar 2017

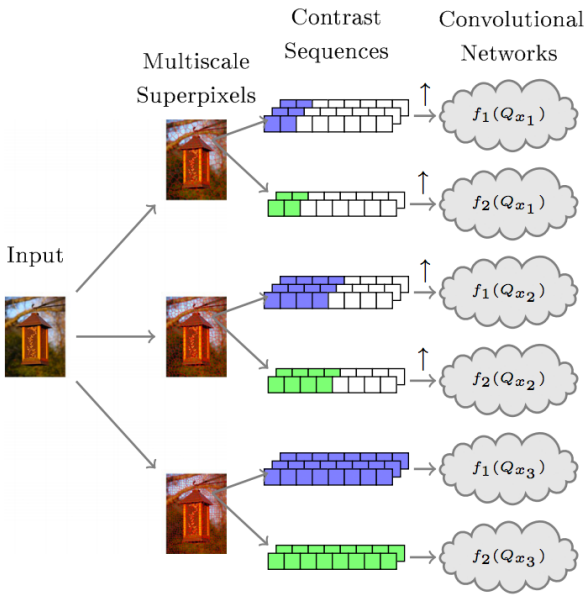
1 Einleitung

- *Superpxiel*: Menge von n Pixeln $S_i = \{t_1, \ldots, t_n\}$, wobei $t_i \in \{1, \ldots, N\}$ jeweils einen Pixel beschreibt und die Menge von S_i räumlich verbunden ist
- Menge von Superpixeln: $S = \{S_1, \ldots, S_m\}$, sodass $S_i \cap S_j = \emptyset$ für alle i, j und $\cup_i S_i = \cup_j t_j$
- Nachbarschaft: $(S_i, S_j) \in \mathcal{N}$, wenn S_i und S_j räumlich verbunden sind
- Vorteile:
  - Superpixel bieten eine Möglichkeit, die Größe des Problems zu minimieren
  - CNNs auf Bildern sind rauschend
  - große Netze auf Bildern mit Megapixeln rechnen langsam
- Nachteil: Superpixel haben einen bestimmten Fehlergrad
- $\Rightarrow$ finde den besten Ausgleich zwischen Größe und Fehlergrad

2 Lernen von Superpixeln

- **SuperCNN:** anstatt eines Bilders wird eine Sequenz von Superpixeln in das CNN gefüttert
- Problem: kontextbezogene Informationen gehen verloren (Methoden wie Superpixel Lattices adressieren dieses Problem, opfern aber Genauigkeit)
- $\Rightarrow$ zwei Kernel sollen Information wiederherstellen:
  1. *Spatial Kernel:* beschreibt Einzigartigkeit der Farben
  2. *Range Kernel:* beschreibt Farbverteilung
- zusätzlich: Multiscale Strukur des Netzes mit *Shared Weights*

- SuperCNN berechnet für individuelles Bild in etwa genauso lange wie klassische CNNs auf Bildern (0.45s)



- Vorteile:
  - benötigt weniger Trainingsdaten
  - Trainingsdaten werden generalisierter genutzt ⇒ Netz fällt es leichter, für unbekannte Bilder Gemeinsamkeiten zu erkennen
  - gleiche oder bessere Performance

3 Features

3.1 Momente

Momente sind in der Bildverarbeitung bestimmte gewichtete Milttelwerte aus den Helligkeitswerten der einzelnen Pixel eines Bildes. Sie werden gewoehnlich so gewaehlt, dass sie gewuenschte Eigenschaften des Bildes widerspiegeln oder gewisse geometrische Interpretationen besitzen. Momente sind hilfreich, um einzelne Objekkte in einem **segmentierten** Bild zu beschreiben.

Momente koennen je nach Wahl von g entweder auf einem Helligkeitsbild oder auf einer Segmentierungsmaske agieren und haben unterschiedliche Bedeutung (gewichtet/ungewichtet).

3.1.1 Nicht-zentrierte Momente

$$M_{ij} = \sum_x \sum_y x^i y^j g(x, y) \tag{1}$$

- **Flaeche:** M_{00}
- **Schwerpunkt:** $\frac{M_{10}}{M_{00}}$ und $\frac{M_{01}}{M_{00}}$

3.1.2 Zentrale Momente

Zentrale Momente sind invariant bezueglich Translationen.

$$\mu_{ij} = \sum_x \sum_y (x - \overline{x})^i (y - \overline{y})^j g(x, y) \tag{2}$$

wobei $\overline{x}$, $\overline{y}$ Schwerpunkt.

- $\mu_{00} = M_{00}$

Informationen ueber die Ausrichtung des Bildes koennen gewonnen werden, indem man zuerst die drei zentralen Momente zweiten Grades verwendet, um eine Kovarianzmatrix zu berechnen

$$\text{cov}[I(x, y)] = \begin{pmatrix} \frac{\mu_{20}}{\mu_{00}} & \frac{\mu_{11}}{\mu_{00}} \\ \frac{\mu_{11}}{\mu_{00}} & \frac{\mu_{02}}{\mu_{00}} \end{pmatrix} = \begin{pmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{pmatrix} \tag{3}$$

Die Eigenvektoren dieser Matrix entsprechen der grossen und kleinen Halbachse der Helligkeitswerte. Die Eigenwerte der Kovarianzmatrix sind

$$\lambda_i = \frac{\mu'_{20} + \mu'_{02}}{2} \pm \frac{\sqrt{4\mu'^2_{11} + (\mu'_{20} - \mu'_{02})^2}}{2} \tag{4}$$

Die *Exzentrizitaet* (engl. *Eccentricity*) des Bildes ist $\sqrt{1 - \frac{\lambda_2}{\lambda_1}}$. Sie ist ein Mass fuer das Groessenverhaeltnis der beiden Hauptachsen. Bei runden Objekten ist sie nahe an 0, bei laenglichen Objekten nahe an 1.

3.1.3 Skalierungsinvariante Momente (normalisiert)

Es koennen Momente η_{ij} mit $i+j \geq 2$ konstruiert werden, die invariant bezueglich Skalierung und Translation sind, indem man das entsprechende zentrale Moment durch das entsprechend skalierte Moment vom Grad 0 teilt.

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{1+\frac{i+j}{2}}} \tag{5}$$

3.1.4 Rotationsinvariante Momente

Es ist weiterhin moeglich, Momente zu konstruieren, die zusaetzlich invariant bezueglich einer Bildrotation sind. Haeufig benutzt wird die *Hu*-Menge invarianter Momente.

- $I_1 = \eta_{20} + \eta_{02}$: Traegheitsmoment um den Schwerpunkt des Bildes, wenn die Helligkeitswerte der Pixel als physikalische Dichte interpretiert werden

3.2 Perimeter

Perimeter beschreibt die Länge der Seitenlinien eines Polygons. Normalerweise ist dies eine recht einfache Aufgabe, denn es müssen lediglich die Längen jeder einzelnen Line addiert werden. Für Pixelpolygone gestaltet sich dies jedoch schwieriger. Das kann gemacht werden, indem die Rahmenpixel bestimmt werden und die Länge der Außenkontur dann um diese Pixel berechnet wird. Das ergibt

$$\text{Anzahl an horizontalen \& vertikalen Pixeln} + \text{Anzahl an Ecken} \cdot \sqrt{2} \tag{6}$$

mit einer Anpassung an den Ecken als Diagonale.

Diese Formulierung hat den Nachteil, dass die berechnete Konturlänge größer ist als die Länge des eigentlichen Polygons. Außerdem ist sie nicht besonders hardwarefreundlich.

Ein alternativer Ansatz ist es, die Linie in der Mitte der Konturpixel zu betrachten. Dafür wird das Polygon zuerst ausgehöhlt. Dann wird zwischen drei unterschiedlichen Möglichkeiten unterschieden:

1. vertikale/horizontale Linien mit Kontribution 1
2. komplettdiagonale Linien mit Kontribution $\sqrt{2}$
3. Halb vertikal/horizontal und halbdiagonale Linien mit Kontribution $\frac{1+\sqrt{2}}{2}$

Dann werden die Pixel entsprechend ihrer Kontribution aufaddiert. Dazu wird die Rahmenmaske *convolved* mit der Maske

$$\begin{pmatrix} 10 & 2 & 10 \\ 2 & 1 & 2 \\ 10 & 2 & 10 \end{pmatrix} \tag{7}$$

Das heißt, wenn wir auf einem Pixel sind wird die direkte Nachbarschaft betrachtet, und falls diese eine 1 besitzt, mit dem jeweiligen Wert in der Matrix aufaddiert. Dann kann einfach entschieden werden, wie die Nachbarschaft eines Pixels aussieht. Ist das Ergebnis 5, 7, 15, 17,

25 oder 27, dann handelt es sich um Fall (1) an diesem Pixel. Ist das Ergebnis 21 oder 33, dann handelt es sich um Fall (2). Ist das Ergebnis 13 oder 23, dann handelt es sich um Fall (3).

Dies funktioniert allerdings nur fehlerfrei, wenn zwei gegenüberliegende Konturen nicht benachbart sind.

3.3 Konvexe Hülle einer Pixelmaske

1. Berechne die x- und y-Koordinaten der 4 Ecken jedes Pixels des Binärbildes
2. Berechne die konvexe Hülle dieser Koordianten
3. Konvertiere diese konvexe Hülle in ein Binärbild mittels `poly2mask`

Es gibt einen Fehler in der obrigen Prozedur. Es können Extrapixel für Diagonalen entstehen. Das ist nicht gut. Würde man den Prozess immer wiederholen, so würde irgendwann ein Rechteck entstehen. Um dies zu beheben, muss Schritt 1 angepasst werden. Statt die Eckpunkte zu sammeln, werden die Zentren der Kanten eines Pixels bestimmt. Das berechnet die richtige Maske bei Diagonalen und es gilt $f(f(x)) = f(x)$

wie heißt das nochmal?

3.4 Polygon-Features

- **Area:** Anzahl an Pixeln im Segment
- **Bounding Box Area:** Flaeche der vertikal/horizontal gelegenen Bounding Box um das Segment
- **Bounding Box Height/Width:** Hoehe bzw. Breite der vertikal/horizontal gelegenen Bounding Box
- **Convex Area:** Anzahl an Pixeln der konvexen Huelle des Segments
- **Local Centroid:** Zentrum/Schwerpunkt des Segments relativ zur Bounding Box
- **Eccentricity:** Groessenverhaeltnis der beiden Hauptachsen einer Ellipse, die das Polygon minimal umschliesst $\in [0, 1]$
- **Major Axis Length:** $4\sqrt{\lambda_1}$
- **Minor Axis Length:** $4\sqrt{\lambda_2}$
- **Perimeter:** Laenge der Seitenlinien des Polygons, kann auch fuer Pixelpolygone berechnet werden (siehe Perimeter Estimator von K. Benkrid mit *4- oder 8-connectivity*)
- **Orientation:** Winkel zwischen der X-Achse und der *Major Axis*
- **Oriented Bounding Box Area:** Das kleinste Rechteck, dass die Region umschliesst.
- **Oriented Bounding Box Axis 1:** Die Laenge der laengeren Seite der Oriented Bounding Box
- **Oriented Bounding Box Axis 2:** Die Laenge der kuerzeren Seite der Oriented Bounding Box

wie bei Pixelpolygonen berechnen? Ist ja eng mit Major Axis verwandt.

Daraus koennen weitere Features ermittelt werden:

- **Extent:** $\frac{\text{Area}}{\text{Bounding Box Area}}$
- **Solidity:** $\frac{\text{Area}}{\text{Convex Area}}$
- **Equivalent Diameter:** Durchmesser eines Kreises mit dem Flaecheninhalt von **Area** $\sqrt{\frac{4 \cdot \text{Area}}{\pi}}$
- **Rectangularity:** $\frac{\text{Area}}{\text{OBB Area}}$
- **Circularity:** $\frac{4\pi \cdot \text{Area}}{\text{Perim} \cdot \text{Perim}}$
- **Compactness:** $\frac{\sqrt{\frac{4}{\pi}\text{Area}}}{\text{OBB Axis 1}}$ (normalisiert Equivalent Diameter)
- **Central moment feature C^2:** $\frac{\mu_{20}+\mu_{02}}{\mu_{00}}$
- **Centrol moment feature C^4:** $\frac{\mu_{40}+\mu_{22}+\mu_{04}}{\mu_{00}}$
- **Elongation:** $\frac{\sqrt{(\mu_{20}-\mu_{02})^2+\mu_{11}^2}}{\mu_{20}+\mu_{02}}$

die naechsten drei gewichtet und/oder ungewichtet?

3.5 Helligkeit-Features

- **Max Intensity:** hoechste Helligkeit im Segment
- **Mean Intensity:** durchschnittliche Helligkeit im Segment
- **Min Intensity:** geringste Helligkeit im Segment
- **Weighted Local Centroid:** Zentrum/Schwerpunkt des Helligkeitssegments relativ zur Bounding Box

3.6 Farb-Features

- **Mean Color:** durchschnittliche Farbe
- **Total Color:** Aufaddierte Farbe aller Pixel im Segment
- **Absolute Difference:** Spannbreite der einzelnen Farbkanaele

3.7 Hole-Features

- **Filled Area**: Anzahl der Pixel, die das Segment enthaelt, wenn Loecher aufgefuellt werden
- **Number of Holes** oder **Euler Number:** Anzahl an Loechern im Segment, definiert als 1 − Number of Holes (*4- oder 8-connectivity*)