Klasse Features

Im folgenden werden für alle Segmente jeweils eine bestimmte Anzahl von Merkmalen berechnet. Logisch gesehen wird hierzu ein zweidimensionaler Speicher der Art `Data[i][j]` benötigt, um den Zugriff auf das j-te Merkmal des i-ten Segments zu realisieren. Um den Datenzugriff aber schneller zu machen, wird statt dessen ein eindimensionales Array verwendet. Wir implementieren die Klasse `Features` also wie folgt:

```
1  class Features{
2  private:
3    int size = 0;      // size of single feature vector
4    int count = 0;     // number of feature vectors
5  public:
6    double* Data = 0; // data buffer
7    Features(){}
8    Features(int s, int c){ Init(s,c); }
9    void Init(int s, int c){
10     if(size!=s || count!=c){
11       size = s;
12       count = c;
13       if(Data!=0){ delete[] Data; }
14       Data = new double[size*count];
15     }
16     memset(Data,0,Size*Count*sizeof(double));
17   }
18   int Size(){ return size; }
19   int Count(){ return count; }
20   double* Vector(int i) { return &Data[i*size]; }
21 };
```

Die äquivalente Ellipse

Die äqivalente Ellipse einer beliebigen Punktwolke hat folgende Gemeinsamkeiten mit der Punktwolke selbst:

- identischer Flächeninhalt (Größe)
- identischer Schwerpunkt (Position)
- identischer Trägheitstensor (Rotationsachsen)

Die Achsen und die Orientierung der Ellipse berechnen sich nach

$$\begin{aligned}
\text{Hauptachse} &= \frac{1}{2}\sqrt{8(m_{20}+m_{02}+\sqrt{(m_{20}-m_{02})^2+4m_{11}})} \\
\text{Nebenachse} &= \frac{1}{2}\sqrt{8(m_{20}+m_{02}-\sqrt{(m_{20}-m_{02})^2+4m_{11}})} \\
\text{Orientierung} &= -0,5\cdot \operatorname{atan2}(2m_{11}, m_{02}-m_{20})
\end{aligned}$$

Dabei ist

$$\text{COV} = \begin{pmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{pmatrix}$$

die Kovarianzmatrix, beschreibt also u.a. die räumliche Ausdehnung in x- und y-Richtung. Die jeweiligen Elemente m_{pq} der Kovarianzmatrix sind die Zentralmomente und berechnen sich nach:

$$m_{pq} = \frac{1}{n} \sum_{i=0}^{n-1} (x_i - \bar{x})^p (y_i - \bar{y})^q$$

Berechnung von Merkmalen

Implementieren Sie die Funktion `featureMoments`:

```
1 void featureMoments(const Image& img, const Image& labelImage,
2   Features& feat, int maxLabel=0);
```

Dabei ist `img` ein Grauwertbild, `labelImage` das Segmentierungsergebnis (Labelbild), `feat` eine Instanz der Klasse `Features` zur Rückgabe der berechneten Merkmal und `maxLabel` die Anzahl der Segmente im Labelbild.
Falls die Anzahl der Segmente unbekannt ist (maxLabel=0), muss diese zuerst in der Funktion `featureMoments` ermittelt werden, um dann den notwendigen Speicher für die Merkmale allokieren zu können.
Berechnen Sie nun für jedes Segment folgende sieben Merkmale:

- Segmentgröße (Anzahl der Pixel)
- Mittlerer Grauwert
- Schwerpunkt (x- und y-Koordinate)
- Achsen und Orientierung der äquivalenten Ellipse