Echtzeitbildverarbeitung SoSe 2015
Prof. Dr.: Udo Frese

Annika Ofenloch - 2992807 - ofenloch@uni-bremen.de
Frank Ihle - 3010158 - fihle@uni-bremen.de
Simon Schirrmacher - 4000884 - simons@informatik.uni-bremen
Noshaba Cheema

Übungsblatt

Abgabe am: 07.05.15

Aufgabe 1: Wer hat mein Tellerchen verrückt? (10 Punkte)

Listing 1: Binarizes and run length codes image

```
void RegionSet::thresholdAndRLE (Mat_<uchar>& image, uchar threshold, int minLength)
{
  // (2P)
  unsigned int length;
  for (unsigned int i = 0; i < image.rows; ++i)
  {
    length = 0;
    for (unsigned int j = 0; j < image.cols; ++j)
    {
      if (image(i, j) > threshold)
      {
        if (length >= minLength)
          rle.push_back(Interval(j - length, j - 1, i));

        image(i, j) = 255;
        length = 0;
      }
      else
      {
        image(i, j) = 0;
        ++length;
      }
    }

  if (length >= minLength)
    rle.push_back(Interval(image.cols - length, image.cols - 1, i));
}
```

Listing 2: Auxiliary routine for unite

```
void RegionSet::pathCompress (Interval* iv)
{
  Interval *root, *buffer;
  root = iv;
  while (root->parent != root)
    root = root->parent;

  while (iv != root)
  {
    buffer = iv->parent;
    iv->parent = root;
    iv = buffer;
  }
}
```

Listing 3: Auxiliary routine for group regions

```
void RegionSet::unite (Interval* iv1, Interval* iv2)
{
  pathCompress(iv1);
  pathCompress(iv2);
  iv1->parent < iv2->parent ? iv2->parent->parent = iv1->parent : iv1->parent->parent = iv2->parent;
}
```

Listing 4: Auxiliary routine for group regions

```
void RegionSet::initialize ()
{
  std::vector<Interval>::iterator it;
  for (it = rle.begin(); it != rle.end(); it++)
    it->parent = &(*it);
}
```

Listing 5: Auxiliary routine for group regions

```
void RegionSet::setRegionIndex ()
{
  std::vector<Interval>::iterator iv;
  iv = rle.begin();
  int regionCtr = 0;
  while (iv != rle.end())
  {
    if (iv->parent == &(*iv))
    {
      iv->region = regionCtr;
      regionCtr++;
    }
    else
      iv->region = iv->parent->region;
    iv++;
  }
}
```

Listing 6: Auxiliary routine for group regions

```
bool RegionSet::touch (Interval* run, Interval* flw)
{
  return run->y == flw->y + 1 && run->xHi >= flw->xLo && flw->xHi >= run->xLo;
}
```

Listing 7: Auxiliary routine for group regions

```
bool RegionSet::ahead (Interval* run, Interval* flw)
{
  return (run->y > flw->y + 1) || (run->y == flw->y + 1 && run->xHi > flw->xHi);
}
```

Listing 8: Finds connected regions in the rle intervals

```
void RegionSet::groupRegions ()
{
  // (3P with functions from pathCompress to setRegionIndex)
  initialize();
  std::vector<Interval>::iterator flw, run;
  flw = run = rle.begin();
  while (run != rle.end())
  {
    if (touch(&(*run), &(*flw)))
      unite(&(*run), &(*flw));
    if (ahead(&(*run), &(*flw)))
      flw++;
    else
      run++;
  }
  setRegionIndex();
}
```

Listing 9: Auxiliary routine for group regions

```
void Region::computeMoments (vector<Region> &region, const RegionSet &decomposition)
{
  // (2P)
  std::vector<Interval> rle = decomposition.rle;
  Interval I;

  for (unsigned int i = 0; i < rle.size(); ++i)
  {
    I = rle[i];
    if (region.size() == I.region)
    {
      region.push_back(Region());
    }
    region[I.region].integral += I.xHi - I.xLo + 1;
    region[I.region].integralX += (I.xHi * (I.xHi + 1) - I.xLo * (I.xLo - 1)) * .5;
    region[I.region].integralY += (I.xHi - I.xLo + 1) * I.y;
    region[I.region].integralXX += (std::pow(I.xHi + .5, 3) - std::pow(I.xLo - .5, 3)) / 3.0;
    region[I.region].integralXY += (I.xHi * (I.xHi + 1) - I.xLo * (I.xLo - 1)) * I.y * .5;
    region[I.region].integralYY += (I.xHi - I.xLo + 1) * (I.y * I.y + 1.0 / 12.0);
  }
}
```

Listing 10: Compute center and inertial axes from the second order moments

```
void Region::computeFeatures()
{
  double dIntegralXX, dIntegralXY, dIntegralYY, eig1, eig2;

  centerX = integralX / integral;
  centerY = integralY / integral;

  dIntegralXX = integralXX / integral - centerX * centerX;
  dIntegralXY = integralXY / integral - centerX * centerY;
  dIntegralYY = integralYY / integral - centerY * centerY;

  eigenDecompositionSymmetric( { {dIntegralXX, dIntegralXY}, {dIntegralXY, dIntegralYY} }, mainAxis,
    eig1, eig2);

  largeLength = 2 * std::sqrt(eig1);
  smallLength = 2 * std::sqrt(eig2);
}
```

Listing 11: Determine label from area and inertial axes

```
void Region::classify()
{
  if (integral >= 5000)
  {
    int ratio = largeLength / smallLength;

    switch (ratio)
    {
      case 1:
        label = "Plate";
        break;
      case 7:
      case 8:
        label = "Spoon";
        break;
      case 10:
      case 11:
        label = "Fork";
        break;
      case 14:
      case 15:
        label = "Knife";
        break;
      default:
        break;
    }
  }
}
```

Aufgabe 2: Spargelzeit (4 Punkte)

Unter Verwendung der Bildverarbeitung soll eine optische Qualitätskontrolle von Spargel durchgeführt werden. Zudem soll unter Berücksichtigung der Spargelqualitätsnorm UNECE-FFV-04 eine Sortierung in die jeweiligen Güteklassen erfolgen.

Zuführung des Spargels

Nachdem der Spargel gewaschen und zugeschnitten wurde, wird er von Mitarbeitern auf ein Fließband abgelegt. Als Annahme wird getroffen, dass der Spargel immer gleich ausgerichtet ist und sich der Spargelkopf oberhalb des Fließbandes befindet. Als nächstes wird der Spargel zu sogenannten Sortierschalen transportiert, indem jeder Spargel separat vom Fließband in eine Sortierschale fällt (siehe Abbildung 1 und 2). Von Mitarbeitern wird nachkontrolliert, ob sich auch wirklich nur ein Spargel in der Schale befindet oder ob ein Spargel daneben gefallen ist. Nicht befüllte Sortierschalen können vorkommen. Sobald sich der Spargel in den Sortierschalen befindet, wird dieser in ein geschlossen System transportiert, wo die Merkmalskontrolle unter Verwendung von Bildverarbeitung durchgeführt wird.

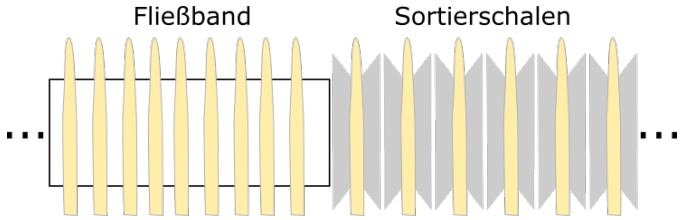

Abbildung 1: Spargel fällt vom Fließband in den Sortierschalen

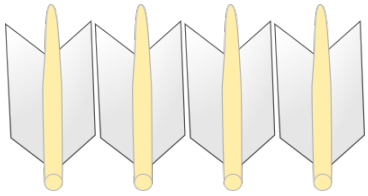

Abbildung 2: Spargel in den Sortierschalen

Kamera und Umgebung

Für die Merkmalskontrolle soll eine Bildaufnahme vom Spargel gemacht werden, wo keine Schattenwürfe oder andere externe Lichteffekte vorhanden sind, da solche Effekte als Schmutz oder Druckstellen interpretiert werden könnten. Die Lichtquelle selbst sollte nah bei der Kamera platziert werden, um einen Schattenwurf zu vermeiden (ggf. mehrere Lichtquellen oder ein Flächenlicht).

Innerhalb des geschlossenen Systems, wo die Bildverarbeitung zum Einsatz kommt, wird eine Kamera verwendet, die sich über den Spargelschalen befindet und senkrecht in Blickrichtung der Schalen ausgerichtet ist. Die Kamera nimmt einen definierten Bereich auf, der acht Sortierschalen umfasst. Demnach können mittels der Bildaufnahme bis zu acht Spargel gleichzeitig geprüft werden.

Fehler- und Merkmalskontrolle

Mit Hilfe der Bildverarbeitung sollen die Merkmale Länge, Durchmesser, Farbe sowie Form des Spargels kontrolliert werden. Es werden zwei Aufnahmen gemacht. Als erstes in der Ausgangsposition und wenn der Spargel um 90 Grad gedreht wurde (Der Spargel bleibt in den Sortierschalen und der herausragende

Teil des Spargels wird auf einer Drehablage aufgelegt, sodass der Spargel gedreht werden kann). Durch die zwei Aufnahmen kann der Spargel von zwei Perspektiven aus betrachtet werden, um Form und Durchmesser zu vergleichen als auch den Spargel gegebenenfalls auf weitere Druckstellen, Verfärbungen und Verschmutzen zu untersuchen.

Auf einer RGB-Bildaufnahme befinden sich acht Spargel, sodass vorerst eine Regionenbildung erfolgen muss. Betrachtet man die Aufnahme als Grauwertbild, unterscheiden sich die Grauwerte von den Sortierschalen (dunkle Grauwerte) von den des Spargels (helle Grauwerte). Demzufolge kann ein automatischer Schwellwert mit Hilfe des Otsu-Agorithmus festgelegt werden, sodass anschließend eine Regionenbildung unter Verwendung des Union-Region-Algorithmus durchgeführt werden kann. Als nächstes wird der Spargel separat in seiner jeweiligen Region betrachtet, um die Qualitätsanforderungen zu kontrollieren. Für die Merkmalskontrolle wird die RGB-Aufnahme verwendet.

Anhand der Spargel-Länge und der runden Spargelspitze wird kontrolliert, ob dieser ganz ist. Die Rundung könnte unter Verwendung eines Kantenerkennungs-Algorithmus geprüft werden. Wenn keine Krümmung (Abgerundeter Kopf) vorhanden ist, wird der Spargel als unvollständig gekennzeichnet und demnach einen entsprechenden Ausgang (z.B. „Fehlerhafter Spargel“) zugeordnet.

Die gesundheitliche und äußere Beschaffenheit des Spargels wird über eine Farbsegmentierung geprüft, indem braune als auch schwarze Verfärbungen betrachtet werden. Sobald der braune beziehungsweise schwarze Farbanteil einen prozentualen Anteil übersteigt, ist der Spargel krank, verschmutzt oder von Schädlingen befallen. Rostflecken und rosafarbige Verfärbungen lassen sich ebenfalls mit einer Farbsegmentierung erkennen. Die Feuchtigkeit lässt sich mit Einschränkungen über eine Farbsegementierung überprüfen, da Wasserreflexionen zu weißen Flecken führen. Demnach kann der weiße Farbanteil näher betrachtet werden. Diese Variante funktioniert gut bei einem grünen Spargel. Bei einem weißen Spargel wird die Erkennung von Wasserflecken schwierig, da bereits ein hoher weißer Farbanteil vorhanden ist. Die Feuchtigkeit muss demnach mit einem zusätzlichen Ansatz geprüft werden, da die Bildverarbeitung hier an ihre Grenzen kommt. Das gleiche gilt für den Geruch, Geschmack und Prallheitsgrad, welche sich nicht mit der Bildverarbeitung kontrollieren lassen.

Ein weiteres Merkmal, das betrachtet werden soll, ist die Schnittfläche am unteren Ende des Spargels. Das abgeschnittene Spargelende muss glatt sein. Hierfür muss geprüft werden, ob ein nahezu rechter Winkel vorhanden ist. Dies sollte bei beiden Bildaufnahmen kontrolliert werden, damit eine zweite Perspektive mit einbezogen wird.

Die Form des Spargels lässt sich kontrollieren, indem von der Spargelmitte aus eine Gerade in Richtung der Hauptträgheitsachsen gezogen wird. Wenn sich ein gewisser Anteil der Gerade nicht im Spargelbereich befindet, ist der Spargel verbogen. Dies sollte ebenso für beide Bildaufnahmen durchgeführt werden, da die Krümmung gegebenenfalls nicht von jeder Perspektive aus sichtbar ist.

Für die Größensortierung wird zusätzlich zur Länge noch der Durchmesser benötigt, der in der Spargelmitte, bezogen auf die Länge, ermittelt werden soll. Hierfür wird eine Gerade in Richtung der Hauptträgkeitsachsen (Breite) bis zur Kante gezogen. Berechnet wird der Durchmesser für beide Bildaufnahmen (erforderlich bei einem ovalen Spargel). Für die Größensortierung (nach dem Durchmesser) wird der größere Durchmesser betrachtet.

Jede Sortierschale bekommt im Verlauf der Qualitätskontrolle einen entsprechenden Ausgang zugeordnet, der sich auf die einzelnen Qualitäts- und Größenklassen bezieht.

Sortierung des Spargels

Der Spargel bleibt vom Übergabebereich des Fließbandes bis zu dem vom Bildverarbeitungssystem zugewiesenen Ausgang in der Sortierschale. Über die Merkmalskontrolle bekommt jede Sortierschale eindeutig einen Ausgang zugewiesen. Bei den Ausgängen handelt es sich um verschiedene Auffangbehälter, wo der Spargel je nach Qualitätsbestimmung hin transportiert wird. Die Auffangbehälter stellen die einzelnen Qualitäts- sowie Größenklassen dar. Sobald die Sortierschale mit dem Spargel den zugehörigen Auffangbehälter erreicht hat, hebt sich die Schale an, sodass der Spargel in die Kiste fällt.

Aufgabe 3: Das runde Dreieck (1 Bonuspunkte)