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 - ncheema@uni-bremen.de

Übungsblatt

Abgabe am: 22.05.15

Aufgabe 1: Cutting-Edge Technology (10 Punkte)

Listing 1: Core routine for \c sobel

```
void sobel1Px (Mat_<ushort>& dstImg, const Mat_<uchar>& srcImg, int x2, int y2)
{
  bool inside = x2 - 1 >= 0 && x2 + 1 < srcImg.cols && y2 - 1 >= 0 && y2 + 1 < srcImg.rows;

  // sobel x filter
  int sX = inside ?
    sobelRound(
    (-srcImg(y2 - 1, x2 - 1) - 2 * srcImg(y2, x2 - 1) - srcImg(y2 + 1, x2 - 1)
    + srcImg(y2 - 1, x2 + 1) + 2 * srcImg(y2, x2 + 1) + srcImg(y2 + 1, x2 + 1)) * 0.125f) : 0;

  // sobel y filter
  int sY = inside ?
    sobelRound(
    (-srcImg(y2 - 1, x2 - 1) - 2 * srcImg(y2 - 1, x2) - srcImg(y2 - 1, x2 + 1)
    + srcImg(y2 + 1, x2 - 1) + 2 * srcImg(y2 + 1, x2) + srcImg(y2 + 1, x2 + 1)) * 0.125f) : 0;

  // filtered pixel
  dstImg(y2, x2) = sobelCode(sX,sY);
}
```

Listing 2: Slow implementation of the sobelX/Y filter

```
void sobel (Mat_<ushort>& dstImg, const Mat_<uchar>& srcImg)
{
  assert (dstImg.size() == srcImg.size());

  // edge detection
  for (int y2 = 0; y2 < dstImg.rows; y2++)
    for (int x2 = 0; x2 < dstImg.cols; x2++)
      sobel1Px(dstImg, srcImg, x2, y2);
}
```

Listing 3: sobel1PxFast as macro function

```
/* sobel1PxFast as macro function
*
* Advantage:
* - makes the functions sobelFast and sobelFastOpenMP twice as fast due to forced inlining
* - less overhead since no type checking
* Disadvantage:
* - because of no type checking, weird things can happen - e.g. when used with expressions such as
   (1+2): #define square(a) a*a
* square(1+2) --> 1+2*1+2 --> 1+2+2 --> 5
* - less readable
*/

#define SOBEL1PXFAST(p, pSrc, sys)\
{\
  int sX = (-(int)pSrc[-sys - 1] - ((int)pSrc[-1] << 1) - (int)pSrc[+sys - 1]\
    + (int)pSrc[-sys + 1] + ((int)pSrc[1] << 1) + (int)pSrc[+sys + 1] + 3) >> 3;\
  \
  int sY = (-(int)pSrc[-sys - 1] - ((int)pSrc[-sys] << 1) - (int)pSrc[-sys + 1]\
    + (int)pSrc[+sys - 1] + ((int)pSrc[+sys] << 1) + (int)pSrc[+sys + 1] + 3) >> 3;\
  \
  *p = sobelCode(sX, sY);\
}
```

Listing 4: Core routine for \c sobelFast

```
void sobel1PxFast (ushort* p, const uchar* pSrc, int sys)
```

```
{
  // uses the sobel filter on one pixel
  SOBEL1PXFAST(p, pSrc, sys);
}
```

```
void sobelFast (Mat_<ushort>& dstImg, const Mat_<uchar>& srcImg)
{
  assert(dstImg.size() == srcImg.size());

  ushort *p, *pEnd, *pLine;
  const uchar *pSrc;
  // step to neighbor
  int sys = srcImg.step[0] / srcImg.step[1];

  pLine = dstImg.ptr<ushort>(0);
  // apply sobel on first line of image - have to be 0 since undefined with a 3x3 filter
  #pragma omp for
  for (int y = 0; y < dstImg.rows; y++)
    pLine[y] = sobel0;

  // apply sobel on second to last - 1 line of image
  #pragma omp for
  for (int y2 = 1; y2 < dstImg.rows - 1; y2++) {
    // current line
    pLine = dstImg.ptr<ushort>(y2);
    // first pixel in line has to be 0 since undefined with a 3x3 filter
    pLine[0] = sobel0;

    // apply sobel filter on the current row, from pixel[1] to pixel[end-2]
    // no parallizing - too much overhead
    for (pSrc = srcImg.ptr(y2) + 1, p = pLine + 1, pEnd = p + dstImg.cols - 2; p < pEnd; p++, pSrc++)
      SOBEL1PXFAST(p, pSrc, sys);

    // last pixel in line has to be 0 since undefined with a 3x3 filter
    pLine[dstImg.cols - 1] = sobel0;
  }

  pLine = dstImg.ptr<ushort>(dstImg.rows - 1);
  // apply sobel on last line of image - have to be 0 since undefined with a 3x3 filter
  #pragma omp for
  for (int y = 0; y < dstImg.rows; y++)
    pLine[y] = sobel0;
}
```

Listing 5: openMP wrapper that calls \c sobelFast in a parallel section

```
void sobelFastOpenMP (Mat_<ushort>& dstImg, const Mat_<uchar>& srcImg)
{
#ifdef NOOPENMP
  std::cerr << "Warning: OpenMP not activated." << endl;
#endif
  assert(dstImg.size() == srcImg.size());

  // distribution on different cores
  #pragma omp parallel
  {
    sobelFast(dstImg, srcImg);
  }
}
```

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.

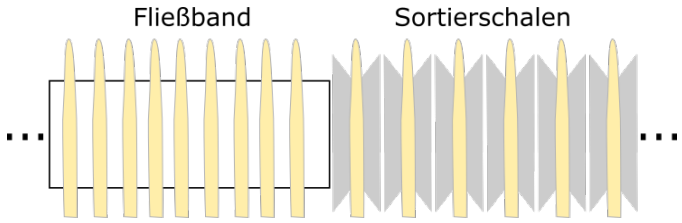


Abbildung 1: Spargel fällt vom Fließband in den Sortierschalen

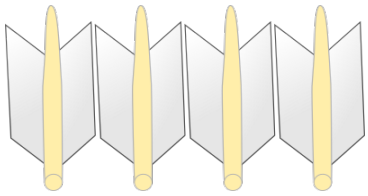


Abbildung 2: Spargel in den Sortierschalen

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, werden zwei Kameras verwendet, die sich über den Spargelschalen und im Winkel von 90° zueinander, in Blickrichtung der Schalenflächen ausgerichtet befinden (s. Abb. 3). Die Kameras nehmen synchron einen definierten Bereich auf, der eine Sortierschale umfasst.

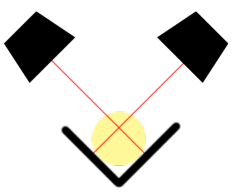


Abbildung 3: Positionierung der Kameras über den Sortierschalen.

Fehler- und Merkmalskontrolle

Mit Hilfe der Bildverarbeitung sollen die Merkmale Länge, Durchmesser, Farbe sowie Form des Spargels kontrolliert werden. Es werden zwei Aufnahmen gemacht.

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) zu denen 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.

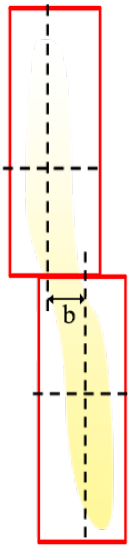
Als nächstes wird der Spargel separat in seiner jeweiligen Region betrachtet, um die Qualitätsanforderungen zu kontrollieren. Für die Merkmalskontrolle wird die RGB-Aufnahme verwendet. Anhand der Spargel-Länge und der runden Spargelspitze wird kontrolliert, ob dieser ganz ist. Die Rundung könnte unter 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 Erkennung selbst erfolgt über einen Flächenvergleich. Entlang der Hauptträgheitsachse sucht man sich den letzten Pixel auf dem Spargel und definiert, ausgehend von diesem Pixel, einen Bereich nach unten (z.B.: eine Spargelbreite). Dann berechnet man den Flächeninhalt des vorliegenden Spargels, von diesem Bereich. Wenn es ein Spargelkopf ist (entspricht in etwa einem Halbkreis) ist die Fläche kleiner als bei einem abgebrochenem Spargel (entspricht in etwa einem Rechteck).

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.

spargel/hauptträgheitsachsen/regionen bild muss verschoben werden.

Der Spargel wird in zwei gleichgroße Bereich aufgeteilt, von denen einzeln die Hauptträgheitsachsen ermittelt werden.(s. Abbildung 4) Anhand des horizontalen Abstands b der vertikalen Achsen kann erkannt werden, ob der Spargel zu krumm ist.



Ein weiteres Merkmal, das betrachtet werden soll, ist die Schnittfläche am unteren Ende des Spargels. Das abgeschnittene Spargelende muss glatt sein.

Hierfür wird vom Schwerpunkt aus, entlang der Hauptträgheitsachse, die Schnittkante am Wurzelende des Spargels gesucht und der letzte Pixel des Spargels definiert. Anhand von diesem gefundenen Pixel wird nun orthogonal zur Hauptträgheitsachse eine Gerade gezeichnet, mit der die Qualität der Schnittkante überprüft werden soll. Nun werden die Flächen ober- und unterhalb (I, II) der gezeichneten Orthogonalen mit Hilfe eines Integrals berechnet (s. Abb. 5). Überschreitet diese Fläche einen bestimmten Schwellwert, ist die Qualitätsanforderung der Schnittkante des Spargels nicht erreicht.

Bei den zwei zuvor beschriebenen Verfahren ist es besonders wichtig, beide erzeugten Bilder auszuwerten, um auszuschließen, dass Qualitätsmängel aus einer Perspektive nicht zu erkennen sind.

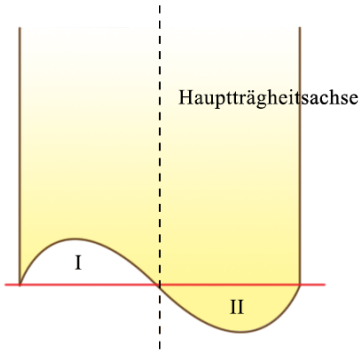


Abbildung 5: Die Schnittkante mit den genannten Flächen, die zur Qualitätseinteilung dienen.

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 (entlang der Breite des Spargels) bis zur Kante gezeichnet. 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, kippt sich die Schale, sodass der Spargel in die Kiste fällt. Anschließend wird der Spargel durch Mitarbeiter verpackt.

Aufgabe 3: Das runde Dreieck (1 Bonuspunkt)

Hier soll auf die Rotation von Regionen eingegangen und deren Vorteil beschrieben werden, wenn eine Region Ω eine Drehsymmetrie besitzt. Zunächst kann die Trägheitsmatrix mit folgender Formel verdreht werden:

$$\begin{pmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{pmatrix}^T \cdot \begin{pmatrix} I'xx & I'xy \\ I'xy & I'yy \end{pmatrix} \cdot \begin{pmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{pmatrix} \tag{1}$$

Ist eine Region drehsymmetrisch, so hat das Einfluss auf die Trägheitsmatrix mit $I'xy = 0$, $I'xx = I'yy$. Das wirkt sich auf (1) wie folgt aus:

$$\begin{pmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{pmatrix} \cdot \begin{pmatrix} I'xx & 0 \\ 0 & I'xx \end{pmatrix} \cdot \begin{pmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{pmatrix} \tag{2}$$

So erhält man:

		$I'xx$	0
		0	$I'xx$
cos α	-sin α	$I'xx$ cos α	$I'xx$ -sin α
sin α	cos α	$I'xx$ sin α	$I'xx$ cos α

$$\begin{pmatrix} I'xx\ cos\alpha & -I'xx\ sin\alpha \\ I'xx\ sin\alpha & I'xx\ cos\alpha \end{pmatrix} \cdot \begin{pmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{pmatrix} \tag{3}$$

und

		cos α	sin α
		-sin α	cos α
$I'xx \cdot cos\alpha$	$-I'xx \cdot sin\alpha$	$I'xx\ (cos^2\ \alpha + sin^2\ \alpha)$	$I'xx \cdot (cos\alpha sin\alpha - sin\alpha cos\alpha)$
$I'xx \cdot sin\alpha$	$I'xx \cdot cos\alpha$	$I'xx\ (cos\alpha sin\alpha - sin\alpha cos\alpha)$	$I'xx\ (cos^2\ \alpha + sin^2\ \alpha)$

mit

$$cos^2\alpha + sin^2\alpha = 1$$

erhält man das Ergebnis:

$$\begin{pmatrix} I'xx & 0 \\ 0 & I'xx \end{pmatrix} \tag{4}$$

Daraus ergeben sich ganz tolle Vorteile.