Bildverarbeitung mit OpenCV

Open Source Computer Vision Library

Johannes Wienke

„Computer Vision Toolkits“
Seminar im Sommersemester 2008
an der Universität Bielefeld

23. Mai 2008

Dieses Tutorial soll eine kurze Einführung in die Programmierung mit der OpenCV — der Open Source Computer Vision Library — bieten. Zunächst werden dazu Zielsetzung und Einsatzgebiete der OpenCV angesprochen. Anschließend wird der Umgang mit der Library anhand einer typischen Bildverarbeitungsaufgabe erklärt. Ausgehend von einem verrauschten und kontrastarmen Bild sollen die Kanten des dargestellten Objekts möglichst solide erkannt werden.

Bemerkungen

Alle Quelltexte und Beispielbilder sind dem Tutorial als Archiv beigefügt. An einigen Stellen wird auf das OpenCV-Wiki verwiesen. Ein Link hierzu findet sich am Ende des Dokuments. Das Wiki ist erste Anlaufstelle für die API-Dokumentation.

1 Was ist OpenCV?

Die Open Source Computer Vision Library (OpenCV) ist eine in C/C++ geschriebene Library, welche ursprünglich von Intel entwickelt wurde, inzwischen aber quelloffen unter einer BSD-Lizenz entwickelt wird. Das OpenCV-Wiki beschreibt die Zielsetzung der Library sehr prägnant mit „OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision. Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking,

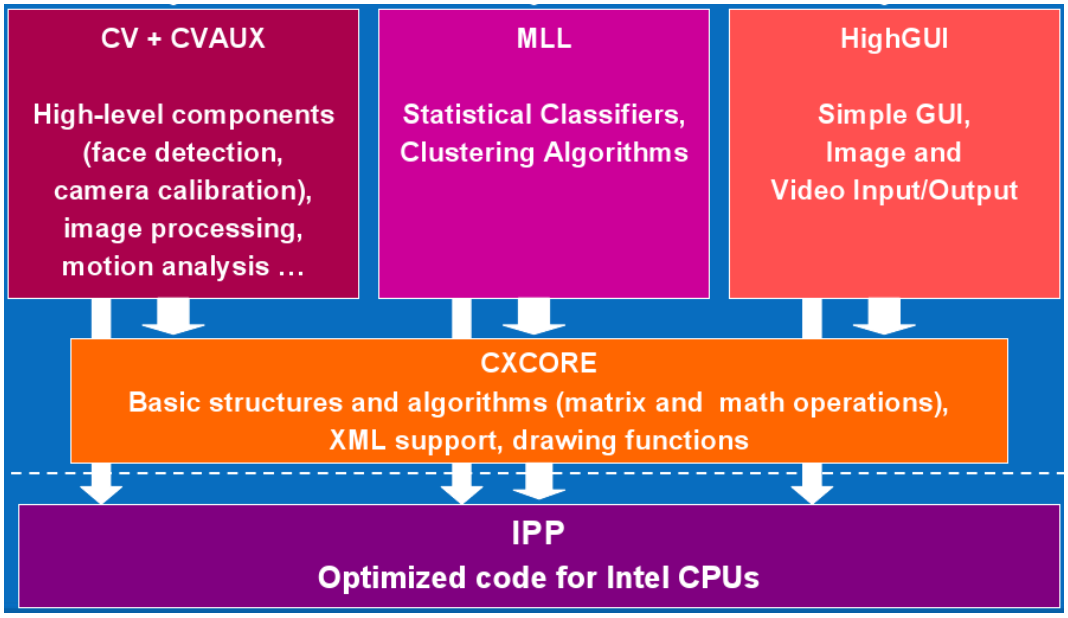

Abbildung 1: Aufteilung von OpenCV in Aufgabengebiete

Ego Motion, Motion Understanding; Structure From Motion (SFM); and Mobile Robotics.“. Die Echtzeitanforderungen spiegeln sich besonders darin wider, dass OpenCV insbesondere auf die Benutzung von Intel-Prozessoren mit Integrated Performance Primitives (IPP) abgestimmt wurde, aber auch ohne diese lauffähig ist.

1.1 Aufbau der Library

Um die genannten Aufgaben zu bewältigen, bietet OpenCV Funktionalitäten auf mehreren Ebenen, die sich in verschiedenen „Paketen“ widerspiegeln. Abbildung 1 beschreibt diese Pakete und ihre Zusammenhänge. Auf unterster Ebene steht die Optimierung für IPP, welche aber für die einfache Benutzung der Library nicht weiter wichtig ist. Interessanter ist das Paket `CxCore`. `CxCore` enthält die grundsätzlichen Datenstrukturen, die für die Arbeit mit OpenCV benötigt werden, insbesondere Punkte, Skalare, Matrizen und Bilder. Es werden aber auch komplexere Datenstrukturen wie Bäume oder Sequenzen angeboten. Des Weiteren enthält `CyCore` die grundlegenden Operationen, um auf diesen Datenstrukturen zu arbeiten. Dies sind u. a. auch Methoden der linearen Algebra und Zeichenfunktionen auf Bildern.

Auf höherer Ebene stehen die Pakete `Cv` (im Wiki auch `CvReference` genannt) und `CvAux`, sowie `Ml` und `Highgui`. `Highgui` bietet Plattform- und Window-Manager unabhängige Möglichkeiten zum Erzeugen einfacher GUIs und zum Laden und Speichern von Bildern und Videos. `Ml` enthält Algorithmen des Machine Learnings, welche im Kontext der Computer Vision eingesetzt werden. `Cv` und `CvAux` enthalten die Algorithmen der Bildbearbeitung und -verarbeitung. `Cv` enthält dabei vor allem Standardalgorithmen, deren Implementierung sehr stabil und effizient ist. `CvAux` hingegen enthält experimen-

telle und veraltete Funktionen.

1.2 Fokus des Tutorials

Dieses Tutorial wird sich vor allem auf die Benutzung der Algorithmen aus `Cv` konzentrieren und dazu die Visualisierung mit der `Highgui` benutzen. Die Konzepte, die anhand von `Cv` erklärt werden, sind in den anderen Bereichen der OpenCV aber genauso gültig.

2 Einstieg: Bilder laden und anzeigen

Zunächst müssen die benötigten Header eingebunden werden. Dies geschieht mittels

```
1  #include <iostream>
2  #include <cv.h>
3  #include <highgui.h>
4
5  using namespace std;
```

`iostream` wird später für die Ausgabe benutzt, `cv.h` und `highgui.h` werden für OpenCV benötigt. Falls Funktionen aus `CvAux` gebraucht werden, muss zusätzlich noch `cvaux.h` inkludiert werden. Um einfach an Ausgabefunktionen etc. zu gelangen, wird der Namespace `std` benutzt.

Anschließend muss (in der main-Methode des Programms) ein Bild geladen werden (hier per Kommandozeilenargument übergeben). Dies erreicht man mit:

```
6  int main(int argc, char *argv[]) {
7
8      IplImage* img = cvLoadImage(argv[1]);
9      if (!img) {
10         cerr << "Could not load image file: " << argv[1] <<
               endl;
11         exit(EXIT_FAILURE);
12     }
```

Der Befehl `cvLoadImage` (Zeile 8) aus `Highgui` kann Bilder in den gängigsten Formaten laden und diese in der von OpenCV benutzten Struktur `IplImage` zur Verfügung stellen. Falls das Laden fehlschlägt, wird `NULL` zurückgegeben und eine Fehlermeldung ausgegeben (ab Zeile 9).

> **Hinweis**
> OpenCV benutzt für Bilder die Datenstruktur `IplImage` aus der Intel Image Processing Library. Daher der für OpenCV sonst ungewöhnliche Prefix `Ipl`.

Im nächsten Schritt wird das Bild mit den Möglichkeiten der `Highgui` angezeigt:

```
13     cvNamedWindow("original", CV_WINDOW_AUTOSIZE);
14     cvShowImage("original", img);
```

Zunächst wird ein Anzeigefenster erzeugt (Zeile 13), welches sich selbst der Größe seines Inhalts anpasst. Anschließend wird mit `cvShowImage` (Zeile 14) das Bild im gerade erzeugten Fenster platziert. Theoretisch wird das Bild so schon im Fenster angezeigt, allerdings endet damit auch sofort die Laufzeit des Programms. Dies lässt sich wie folgt verhindern.

```
15      cvWaitKey(0);
```

Der Befehl `cvWaitKey` (Zeile 15) startet einen Main-Loop, der solange läuft, bis eine Taste betätigt wird. In dieser Zeit wird dann auch das Bild angezeigt.

Hinweis
Ein Blick in das OpenCV-Wiki zu `cvShowImage` zeigt, dass die Funktion als Argument für das anzuzeigende Bild keinen Pointer vom Typ `IplImage` erwartet, sondern einen `CvArr`-Pointer. Dieser Datentyp wird in OpenCV als allgemeiner Datentyp für Operationen, die mit Matrix-ähnlichen Daten (es gibt z. B. auch Matrizen in OpenCV) umgehen können, benutzt. Zur Laufzeit wird dann überprüft, ob die Funktion wirklich mit den Daten arbeiten kann.

Zu guter Letzt müssen noch Aufräumarbeiten durchgeführt werden.

```
16      cvDestroyWindow("original");
17      cvReleaseImage(&img);
18
19      return EXIT_SUCCESS;
20  }
```

Da es sich um C/C++ handelt, müssen alle (implizit über OpenCV) angeforderten Speicherbereiche explizit wieder freigegeben werden. Für Fenster geschieht dies mit der Funktion `cvDestroyWindow` und für Bilder mit der Funktion `cvReleaseImage`.

Hinweis
Wie leicht zu erkennen ist, werden Fenster in der OpenCV Highgui ausschließlich per Namen angesprochen. OpenCV beschwert sich nicht über unbekannte Namen beim Aufruf von `cvDestroyWindow`. Der Name ist gleichzeitig auch der Fenstertitel.

Das Programm (gespeichert in der Datei `sample.cpp`) kann nun mit den Befehlen

```
g++ sample.cpp -I /usr/include/opencv -lcv -lhighgui -o sample
./sample sax1.ppm
```

übersetzt und ausgeführt werden, so dass ein Bild angezeigt wird. Der Quellcode im Archiv enthält zusätzlich noch eine Überprüfung des Kommandozeilenarguments.

Hinweis
Falls Funktionen aus `CvAux` benutzt werden, muss dem Linker auch diese Library mittels `-lcvaux` mitgeteilt werden.

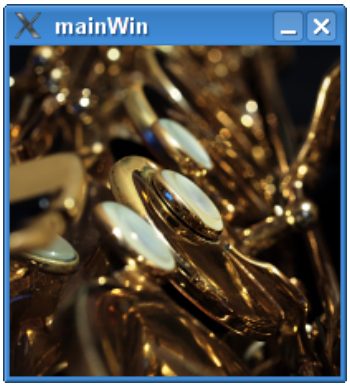

Abbildung 2: Ausgabe des ersten OpenCV-Programms

3 Bildverarbeitung mit OpenCV

Die Möglichkeiten zum einfachen Erstellen von GUIs mit OpenCV sind z. B. fürs Prototyping sehr praktisch, aber lange nicht Kern von OpenCV. Wichtiger sind u. a. die Bildverarbeitungsfunktionen der OpenCV, welche in diesem Abschnitt eingeführt werden. Dies geschieht anhand einer für die Bildverarbeitung typischen Aufgabe: Aus dem zum Tutorial mitgelieferten Bild `verrauscht.tiff` sollen möglichst gut die Kanten des Lampengehäuses extrahiert werden. Das Programm kann nach allen Zwischenschritten mit den gleichen Befehlen wie weiter oben kompiliert und ausgeführt (allerdings mit dem Argument `verrauscht.tiff`) werden, um so einen Eindruck für die Zwischenergebnisse zu gewinnen.

3.1 Filter

3.1.1 Histogrammausgleich

OpenCV stellt viele Standardfilter und Operationen der Bildverarbeitung zur Verfügung. Da das Beispielbild ein schlechtes Kontrastverhältnis aufweist, sollte ein Histogrammausgleich durchgeführt werden. Dazu wird folgender Code vor die Ausgabe im eben erstellten Programm eingefügt:

```
1       IplImage *binaryImage = cvCreateImage(cvGetSize(img),
           IPL_DEPTH_8U, 1);
2       cvSplit(img, binaryImage, NULL, NULL, NULL);
3       IplImage *normalized = cvCreateImage(cvGetSize(img), img
           ->depth, 1);
4       cvEqualizeHist(binaryImage, normalized);
5
6       cvNamedWindow("normalized", CV_WINDOW_AUTOSIZE);
7       cvShowImage("normalized", normalized);
8   }
```

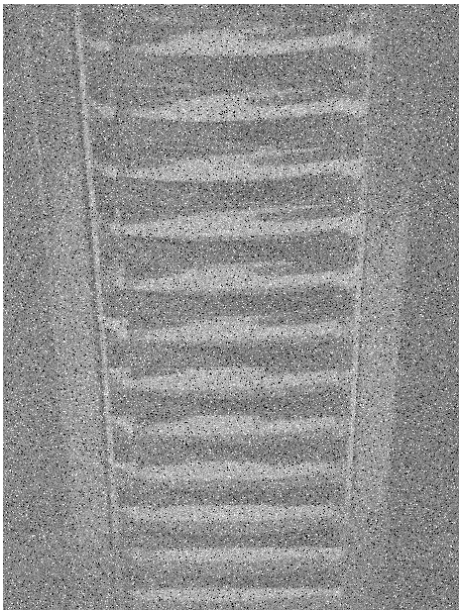

Abbildung 3: Beispielbild `verrauscht.tiff` mit Rauschen und schlechtem Kontrast

Der Histogrammausgleich erfolgt über die Funktion `cvEqualizeHist` (Zeile 4). Diese Funktion arbeitet nur auf Grauwertbildern mit einem Farbkanal. Leider hat `cvLoadImage` das Bild (obwohl es schon ein Grauwertbild ist) mit drei Farbkanälen geladen. Die Anzahl der Farbkanäle kann über `img->nChannels` abgerufen werden. Um ein Bild mit nur einem Kanal zu erzeugen, werden die Kanäle des Ursprungsbildes mit `cvSplit` auf Bilder mit jeweils einem Kanal gesplittet. Der relevante Kanal mit dem Grauwertbild wird dadurch in `binaryImage` gespeichert. `binaryImage` muss ebenso wie das Ausgabebild für `cvEqualizeHist` vorher mit `cvCreateImage` angelegt werden.

Hinweis
`cvSplit` erwartet neben dem Eingabebild (erstes Argument) noch vier weitere Argumente. Dies sind die einkanaligen Bilder, auf die das Ursprungsbild aufgeteilt werden soll. Sind einige dieser Argumente `NULL`, wird der entsprechende Kanal verworfen. Vier Argumente reichen aus, da OpenCV Bilder mit maximal vier Kanälen verarbeiten kann.

Hinweis
Der Aufruf von `cvCreateImage` erwartet neben der Bildgröße (ein Struct vom Typ `CvSize`) auch die Farbtiefe und die Anzahl der Kanäle. Die Farbtiefen-Konstanten werden im OpenCV-Wiki erklärt.

Hinweis
Viele Funktionen der OpenCV können nicht mit Bildern aller Farbtiefen und Kanalzahlen

umgehen. Teilweise hilft die Dokumentation im Wiki weiter, teilweise muss aber leider auch einfach ausprobiert werden.

3.1.2 Gaussfilter und Kantenfindung

Das Rauschen im Bild kann z. B. mit einem Gaussfilter reduziert werden. Anschließend kann versucht werden die Kanten zu extrahieren. Hierfür bietet OpenCV u. a. eine Implementierung des Canny-Algorithmus an. Da die beiden Schritte analog zum Histogrammausgleich ablaufen, wird der Quellcode nicht weiter beschrieben. Die Parameter der Funktionsaufrufe können dem Wiki entnommen werden.

```
9     IplImage *gauss = cvCreateImage(cvGetSize(img), img->
          depth, 1);
10    cvSmooth(normalized, gauss, CV_GAUSSIAN, 13, 13);
11
12    cvNamedWindow("gauss", CV_WINDOW_AUTOSIZE);
13    cvShowImage("gauss", gauss);
14
15    IplImage *canny = cvCreateImage(cvGetSize(img), img->
          depth, 1);
16    cvCanny(gauss, canny, 40, 130);
17
18    cvNamedWindow("edges", CV_WINDOW_AUTOSIZE);
19    cvShowImage("edges", canny);
```

3.1.3 Morphologische Operationen

Die Ergebnisse des Kantenbildes sind noch lange nicht überzeugend. Evtl. lassen sich einige der entstandenen kleinen Lücken mittels Closing schließen. Diese morphologische Operation kann mit OpenCV wie folgt durchgeführt werden:

```
20    IplImage *temp = cvCreateImage(cvGetSize(img),
          IPL_DEPTH_8U, 1);
21    IplImage *closed = cvCreateImage(cvGetSize(img),
          IPL_DEPTH_8U, 1);
22    IplConvKernel* element = cvCreateStructuringElementEx(3,
          3, 1, 1, CV_SHAPE_RECT);
23    cvMorphologyEx(canny, closed, temp, element,
          CV_MOP_CLOSE);
24    cvReleaseStructuringElement(&element);
25    cvReleaseImage(&temp);
26
27    cvNamedWindow("closed", CV_WINDOW_AUTOSIZE);
28    cvShowImage("closed", closed);
```

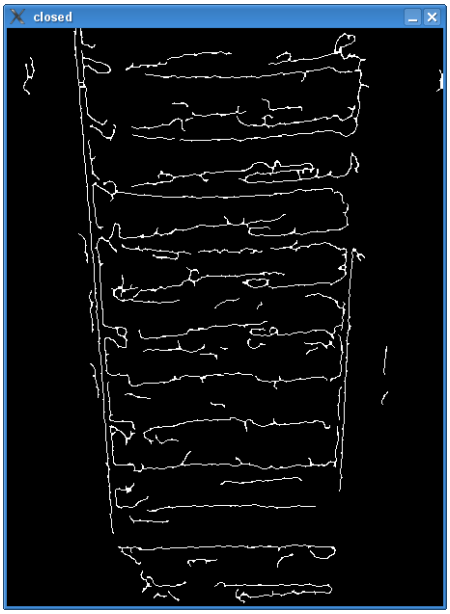

Abbildung 4: Ergebnis der Kantenextraktion

Morphologische Operationen werden in OpenCV vor allem mit `cvMorphologyEx` (Zeile 23) durchgeführt. Diese Funktion benötigt neben Ein- und Ausgabebild auch noch ein temporäres Bild zum Arbeiten (erzeugt in Zeile 20) und ein Argument vom Typ `IplConvKernel`. Dies ist das strukturierende Element der morphologischen Operation und wird mit `cvCreateStructuringElementEx` (Zeile 22) angelegt und muss mit `cvReleaseStructuringElement` wieder aufgeräumt werden (Zeile 24).

3.2 Letzte Aufräumarbeiten

Alle erzeugten Bilder müssen auch wieder aufgeräumt werden, genauso wie die angezeigten Fenster. Für die Bilder muss dies jeweils einzeln mit einem Aufruf von `cvReleaseImage` geschehen. Anstatt alle Fenster einzeln von Hand aufzuräumen, kann dies auch in einem Schritt mittels `cvDestroyAllWindows` geschehen (der Aufruf von `cvDestroyWindow` kann also hiermit ersetzt werden).

Wenn alles geklappt hat, sollte nach dem Kompilieren und Ausführen neben den Zwischenergebnissen das Bild 4 angezeigt werden.

Der gesamte Quellcode zu diesem Beispiel ist im Archiv als `solution.cpp` enthalten.

4 Zusammenfassung

Das Tutorial hat einen kleinen Teil der Bildverarbeitungsfunktionen von OpenCV angerissen und deren Verwendung demonstriert. Hieraus lassen sich jedoch einige allgemeine Punkte für die Verwendung von OpenCV ableiten:

- Alle Funktionen von OpenCV beginnen mit dem Prefix `cv`.
- Die meisten Datentypen beginnen mit dem Prefix `Cv`. Ausnahmen bilden die aus der Image Processing Library übernommenen Datentypen.
- Eigentlich alle Datentypen werden über Factorymethoden erzeugt und müssen nach der Verwendung von Hand wieder aufgeräumt werden.
- Ein weit verbreitetes Schema für Funktionsaufrufe von Algorithmen ist `cvFunktionsname(<source>, <dest>, [Parameter])`,
  wobei `source` das zu verarbeitende Bild und `dest` das Ergebnisbild bezeichnet.

5 Referenzen und Links

- http://www.intel.com/technology/computing/opencv/index.htm
- http://opencvlibrary.sourceforge.net — das OpenCV-Wiki
- http://www.cs.iit.edu/ agam/cs512/lect-notes/opencv-intro/index.html
- http://tech.groups.yahoo.com/group/OpenCV/
- http://fsa.ia.ac.cn/files/OpenCV_China_2007June9.pdf

Aufgabe

Erweitere das entwickelte Beispiel um eine Ausgabe, die zusammenhängede Kantenstücke (Konturen) des Lampengehäuses farblich voneinander abgegrenzt darstellt. Die hierfür notwendingen Funktionen sind im OpenCV-Wiki noch nicht hinreichend dokumentiert. Folgender Link enthält eine vollständigere Dokumentation:
http://www710.univ-lyon1.fr/ bouakaz/OpenCV-0.9.5/
docs/ref/OpenCVRef_ImageProcessing.htm

Hinweis
Falls eine der benutzten Funktionen einen Pointer auf einen `CvSeq`-Pointer erwartet, musst du dich nicht selbst um die Allozierung dieser Datenstruktur kümmern. Es reicht aus, die Adresse eines mit 0 initialisierten `CvSeq`-Pointers zu übergeben.

Hinweis
Farbwerte lassen sich unter anderem mit dem Makro `CV_RGB` erzeugen.