Karlsruher Institut für Technologie
Andreas Bihlmaier
andreas.bihlmaier@gmx.de

KIT – Karlsruher Institut für Technologie
Fakultät für Informatik
Institut für Prozessrechentechnik und Robotik (IPR)

Entwicklung einer Visualisierung für Bildverarbeitungssoftware

Praxis der Softwareentwicklung WS 2013

Lastenheft

Revision: 1.0

Inhaltsverzeichnis

I Projektbeschreibung

Im Rahmen des Projekts soll eine C++ Bibliothek zur Visualisierung von Bildverarbeitungssoftware entwickelt werden. Die Bibliothek soll dabei auf Qt als GUI Framework aufbauen und für die Verwendung mit OpenCV angepasst sein. Einzelschritte der Bildverarbeitung im Sinne von OpenCV Funktionen - wie Filter, morphologische Operationen, Feature Matches und Transformationen - sind in einem GUI zu visualisieren. Über ein passend entworfenes API soll diese Funktionalität einfach in bestehenden Code eingebunden werden können und im deaktivierten Zustand das Laufzeitverhalten nicht negativ beeinflussen. Die Auswirkung der verschiedenen Bildoperatoren soll interaktiv auf das gewünschte Verhalten hin untersucht werden können. Zweck der Bibliothek ist die interaktive, visuell unterstützte Entwicklung von OpenCV Bildverarbeitungsketten sowie Werkzeuge zum high-level Debugging der verschiedenen Verarbeitungsschritte.

I.1 Ausgangsproblemstellung und erwartete Verbesserung

Während der Entwicklung von Bildverarbeitungsalgorithmen tritt wiederholt das Problem auf Einzelschritte auf ihre gewünschte Funktion hin zu untersuchen. Durch den Mangel an bereitstehenden Werkzeugen - insbesondere Visualisierungen - für diese Aufgabe, muss der Entwickler auch diese ständig neu erfinden und programmieren.

Die durch das vorliegende Projekt gewünschte Verbesserung besteht in der Bereitstellung, von Visualisierungen für unterschiedliche Arbeitsschritte der Bildverarbeitung auf C++ API Ebene. Anstatt mehrere Zeilen Code zur Erzeugung von statischen, visuellen Debugausgaben aufzuwenden, soll dem Programmierer ermöglicht werden durch einfache Methodenaufrufe das Ergebnis des gewünschten Schrittes in einer GUI interaktiv zu bewerten.

Die zugrundeliegenden Bibliotheken, Qt für die GUI Komponenten und OpenCV für die Bildverarbeitung, werden im Folgenden kurz beschrieben.

I.2 Qt

Qt ist eine C++-Klassenbibliothek für die plattformübergreifende Programmierung grafischer Benutzeroberflächen. Neben der Entwicklung grafischer Benutzeroberflächen bietet Qt umfangreiche Funktionen zur Internationalisierung, Netzwerk- und Interprozesskommunikation, Datenbankfunktionen sowie XML-Unterstützung an und ist für verschiedene Betriebssysteme bzw. Grafikplattformen erhältlich. Qt verwendet einen Präprozessor, genannt MOC (meta object compiler), womit C++ um Fähigkeiten bereichert wird, die im Sprachstandard nicht enthalten sind, beispielsweise Signale und Slots sowie Introspektion. Der so erzeugte Code folgt dem C++-Standard, so dass er mit handelsüblichen Compilern übersetzt werden kann. Neuere Qt Versionen stehen unter der LGPL.[1][2]

I.3 OpenCV

OpenCV ist eine freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und maschinelles Sehen. Sie ist für die Programmiersprachen C und C++ geschrieben und steht als freie Software unter den Bedingungen der BSD-Lizenz. Die Stärke von OpenCV liegt in ihrer Geschwindigkeit und in der großen Menge der Algorithmen aus neuesten Forschungsergebnissen.[3] Die OpenCV Bibliothek besteht aus mehr als 2500 für diverse Zielplattformen optimierten Algorithmen. Darunter enthalten sind Algorithmen zur Gesichtsdetektion und -erkennung, Objekterkennung, Aktionsklassifikation in Videos, Tracking von Kamerabewegungen, Tracking von bewegten Objekten, Extraktion von 3D Objektmodellen, Erzeugung von 3D Punktwolken mit Stereokameras, Image Stitching um aus Einzelaufnahmen ein großes Panorama zu erzeugen, Mustererkennungs- und indizierungsmethoden, sowie klassische Filteralgorithmen und schnelle Merkmalserzeugung und -matching. OpenCV wird weltweit eingesetzt und besitzt dabei eine Community von ungefähr 47.000 Entwicklern und professionellen Nutzern. Open-

[1] Wikipedia: *Qt (Bibliothek) — Wikipedia, Die freie Enzyklopädie*.
[2] *Qt Project*.
[3] Wikipedia: *OpenCV — Wikipedia, Die freie Enzyklopädie*.

CV wurde mehr als sieben millionenfach heruntergeladen.[4]

II Systemübersicht

Code:

```
// Find ORB features in two images
ImageFeatures features1, features2;
OrbFeaturesFinder orbFinder(...);
orbFinder(inputImg1, features1);
orbFinder(inputImg2, features2);

// Match found features in both images
vector<DMatch> matches;
Matcher matcher(...);
matcher.match(features1, features2, matches);

// For Development only (does nothing in release mode):
// Interactively debug matches and inspect parameters
cv::visualize::showMatches(inputImg1, inputImg2, matches);
```

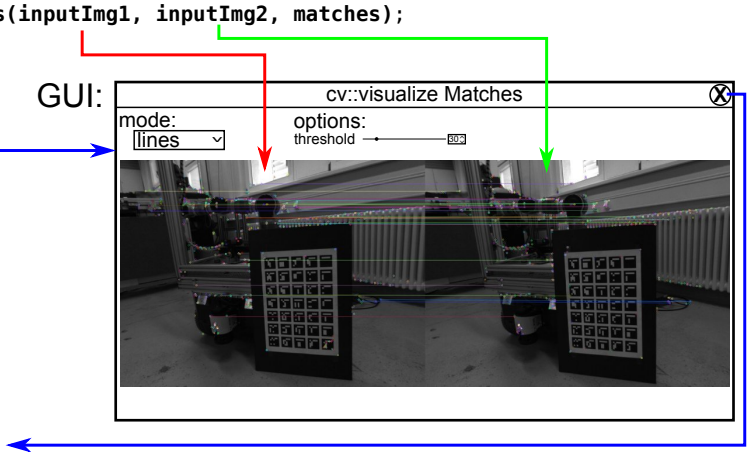

Abbildung 1: Übersicht des zu entwickelnden Systems bestehend aus API Aufrufen im Quellcode und einer GUI, welche zur Laufzeit des Programms ausgeführt wird. Die Pfeile verdeutlichen Kontroll- und Datenfluss.

III Zu entwicklende Softwarelösung

Die zu entwicklende Bibliothek soll den Standards für objektorientierte Programmierung (OOP) entsprechen und in C++ entwickelt werden. Aufbau und Codekonventionen sollen sich nach denjenigen in OpenCV und Qt, in dieser Prioritätsordnung, richten. Die Zielplattform ist Linux,

[4] *OpenCV.org*.

jedoch soll soweit möglich die plattformunabhängigkeit der zugrundeliegenden Bibliotheken bewahrt werden, d.h. plattformunabhängige Lösungen sind plattformspezifischen vorzuziehen. Als Build-System soll CMake zum Einsatz kommen und dessen Automatismen zur Programmerzeugung - z.B. `find_package` - werden.[5] Das Integrationsziel der entwickelten Software ist als ein reguläres Modul in OpenCV einzugehen. Diese Zielvorgabe soll als Leitfaden dienen, muss jedoch nicht tatsächlich erreicht werden. In jedem Fall empfiehlt sich während der Entwicklung informellen Kontakt mit der OpenCV Entwicklercommunity aufzunehmen[6], um die Randbedingungen und Schwerpunkt der initialen Bibliothek richtig zu setzen, Akzeptanz sowie Nutzen der Community zu steigern und um künftigen Erweiterungen den Weg zu bereiten.

IV Produkteinsatz

Die Software soll zunächst im universitären Forschungsumfeld des beauftragenden Institutes eingesetzt werden. Später kann der Nutzerkreis potentiell auf alle OpenCV Benutzer ausgedehnt werden, welche OpenCV mit Qt Unterstützung kompiliert haben.

V Produktfunktionen

V.1 Bewertung der gewünschten Produktfunktionen

! Musskriterien

\+ Wunschkriterien

V.2 Gewünschte Produktfunktionen

- API
  - Abzudeckende Module

[5] *CMake*.
[6] `http://opencv.org/contribute.html`

      - ! imgproc/Image Filtering: dilate, erode, morphologyEx, Sobel,
      - ! imgproc/Miscellaneous Image Transformations: threshold, adaptiveThreshold, floodFill
      - + imgproc/Histograms: calcHist
      - + imgproc/Feature Detection: Canny, HoughCircles
      - ! features2d: KeyPoint, DMatch
      - + stitching
      - + ocl
    - – Parametrisierbarkeit
      - ! Auswahl der Visualisierung für Operationstyp
      - + Optionale Parameter für Einstellungen der Visualisierungen
    - ! Globale Auswahl zwischen Debug und Release Modus
    - + Lokale Auswahl Debug/Release
    - + Optionale nicht-blockierende Aufrufe für Streaming
- GUI
    - ! Eine Visualisierung pro Operation
    - ! Drei Visualisierungen für features2d/DMatch
    - ! Zoomfunktion
    - + Hohe Zoomstufen mit Zusatzinformationen (z.B. Pixelwerte)
    - + Parameter für Visualisierungen
    - + Weitere Visualisierungen
    - + Permanente GUI mit Historie
    - + Möglichkeit Operation mit geänderten Parametern erneut anzuwenden
    - + Datenfluss von GUI zu Code sofern sinnvoll

+ Optionale Ausnutzung von mehreren Bildschirmen

+ Interaktive Überlagerung von Zusatzinformationen (Mouse-Over)

VI Produktleistungen

! Die GUI soll schnell starten und interaktiv bedienbar sein

+ Beim Streaming sollen die Leistungseinbußen bei einfachen Visualisierungen niedrig sein

! Im Release Modus sollen keine wesentlichen Leistungseinbußen messbar sein

+ Flexibler Umgang mit unterschiedlichen Bildschirm- und Bildauflösungen

+ Integration in OpenCV Test Framework

VII Qualitätsanforderungen

! Keine signifikanten Speicherlecks

! Erweiterbarkeit um zusätzliche OpenCV Operationen und Visualisierungen

! Modularer Aufbau (API und GUI)

! Einhaltung der OpenCV und Qt Konventionen

! Ausführliche Dokumentation der API und des GUI

+ Dokumentation des internen Codes mit Werkzeug

+ OpenCV geeigneter Aufbau des Build-Systems

+ Abdeckung durch Tests

VIII Systemumgebung

- Ubuntu 12.04
- Qt 4.8 oder neuer
- OpenCV 2.3.1 (bevorzugt 2.4.6 oder neuer)
- CMake 2.8.7 oder neuer
- C++ (Standard C++03 oder C++11)
  - g++ Compiler (Version 4.6 oder neuer, bevorzugt 4.8)

Literatur

Blanchette, Jasmin und Mark Summerfield: *C++ GUI Programming with Qt4*, [2]2008.

CMake, [Online; Stand 4. November 2013], 2013, URL: http://cmake.org/.

Grimm, Rainer: *C++11*, [1]2012.

Laganière, Robert: *OpenCV 2 Computer Vision Application Programming Cookbook*, Birmingham [1]2011.

OpenCV.org, [Online; Stand 4. November 2013], 2013, URL: http://opencv.org/.

Qt Project, [Online; Stand 4. November 2013], 2013, URL: http://qt-project.org.

Wikipedia: *OpenCV — Wikipedia, Die freie Enzyklopädie*, [Online; Stand 4. November 2013], 2013, URL: http://de.wikipedia.org/w/index.php?title=OpenCV&oldid=121932015.

Wikipedia: *Qt (Bibliothek) — Wikipedia, Die freie Enzyklopädie*, [Online; Stand 4. November 2013], 2013, URL: http://de.wikipedia.org/w/index.php?title=Qt_(Bibliothek)&oldid=123854287.

Wolf, Jürgen: *Grundkurs C++* (Galileo Computing), Bonn [2]2013.