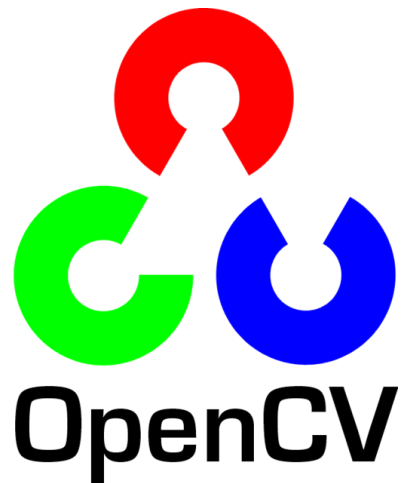


CVVisual – Eine Visualisierung für OpenCV

Pflichtenheft

Johannes Bechberger, Erich Bretnütz,
Nikolai Gaßner, Raphael Grimm,
Clara Scherer, Florian Weber

25. November 2013



OpenCV ist eine OpenSource Bilderkennungs-Bibliothek. Sie wurde im Jahre 2000 der breiteren Öffentlichkeit vorgestellt, 6 Jahre später erschien Version 1.0.

Seitdem hat sich das Projekt stetig weiterentwickelt - sie ist heute quasi der Standard unter den freien Bilderkennungs-Bibliotheken. Sie umfasst heutzutage Algorithmen für zum Beispiel einfache Bilderkennung, Gesichtserkennung, Bewegungsverfolgung und vieles mehr. Die Bibliothek ist auf Performance optimiert und findet auch deshalb heute Einsatz in vielen Bereichen, wie etwa der Augmented Reality auf mobilen Geräten oder an Universitäten.

Wer jedoch damit arbeitet, steht zurzeit noch vor einem Problem:

Es existiert keine richtige Debug-Visualisierung für OpenCV. Das bedeutet, dass es augenblicklich keine Bibliothek (und auch kein Werkzeug) gibt, mit welchem man sich adäquat visualisieren lassen kann, was zum Beispiel eine lange Liste von Matches mit den zwei dazugehörigen Bildern zu tun hat, oder welche Auswirkung ein bestimmter Filter hat. Deswegen fangen Entwickler oft an, eigene Lösungen für dieses Problem zu entwickeln, die in der Hauptsache aus dem einfachen Speichern der Bilder oder der Verwendung primitiver Methoden von OpenCV (wie etwa `'imshow()'`, das einfach nur ein Bild anzeigt) in Kombination mit schnell zusammengeschusterten Zeichenroutinen bestehen.

Besonders für Neulinge stellt dieses weitestgehende Fehlen von Visualisierungsmöglichkeiten zu Debugzwecken eine große Hürde dar.

Unser Ziel als PSE-Team ist es, daran etwas mit unserer Arbeit an einer Open Source-Visualisierung für OpenCV zu verändern.

Inhaltsverzeichnis

1	Produktfunktionen	4
2	Produkteinsatz	4
3	Produktumgebung	4
4	Funktionale Anforderungen	5
4.1	API <i>Pflicht</i>	5
4.2	API <i>Optional</i>	5
4.3	GUI <i>Pflicht</i>	6
4.4	GUI <i>Optional</i>	6
4.5	Mögliche Visualisierungen	6
5	Nichtfunktionale Anforderungen	7
5.1	Produktleistungen <i>Pflicht</i>	7
5.2	Produktleistungen <i>Optional</i>	7
5.3	Qualitätsanforderungen <i>Pflicht</i>	8
5.4	Qualitätsanforderungen <i>Optional</i>	8
6	Produktdaten	8
7	Systemmodell	9
8	Bedienoberfläche	10
8.1	Skizzen	10
9	Testfälle und Testszenarien	18
9.1	Tests für Filterview	18
9.2	Tests für Matchview	18
9.3	Beispielhafter Ablauf	18
10	Abgrenzungskriterien	19
11	Entwicklungsumgebung	19
11.1	Quellcode	19
11.2	Dokumentation	19
12	Lizenz	19
12.1	Quellcode	20
12.2	Dokumentation	20
13	Glossar	21
13.1	Allgemein	21
13.2	OpenCV	23
14	Literatur	23

1 Produktfunktionen

Unser Produkt ist ein Debug-Werkzeug für diverse OpenCV-Funktionalität. Hierzu werden die Ergebnisse von Filteroperationen verwendet, um die Auswirkungen des Filters auf das Ursprungsbild zu visualisieren.

Um das Debuggen zu erleichtern und Codeänderungen im Anschluss überflüssig zu machen, ist die Funktionalität dabei so implementiert, dass pro Translation Unit der Debug-Modus sowohl während des Kompilervorgangs als auch zur Laufzeit (de-)aktiviert werden kann. Bei der Deaktivierung während des Kompilierens wird hierbei die Programmlaufzeit in keiner Weise negativ beeinflusst.

Bei der Verwendung in Programmen mit mehreren Threads ist zwar mit eingeschränktem Komfort zu rechnen, die prinzipielle Funktionalität an sich bleibt dennoch unbeeinträchtigt.

Zur Verwendung stellen wir für die debugbaren OpenCV-Funktionalitäten Funktionen bereit, welche eine graphische Darstellung des Filters zu einem großen Debug-Hauptfenster hinzufügen (pro Thread ein Hauptfenster).

Je nach View stellen wir beispielsweise die Matches zwischen zwei Bildern mit Pfeilen dar. Hierbei gibt es auch eine Möglichkeit, die Darstellungen selbst zu filtern, beispielsweise indem nur Pfeile zwischen Unterschieden, die einen gewissen Schwellwert überschreiten, gezeichnet werden.

2 Produkteinsatz

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

3 Produktumgebung

Nach Möglichkeit alle Plattformen auf denen moderne Versionen von OpenCV und Qt5 laufen, sowie ein C++11-Compiler existiert.

4 Funktionale Anforderungen

Erklärungen zu den OpenCV spezifischen Begriffen werden im Glossar gegeben.

4.1 API Pflicht

4.1.1 Allgemein

/FA0100/ Globale Auswahl zwischen Debug- und Release-Modus

/FA0200/ Auswahl der Visualisierung für jeden Operationstyp

4.1.2 Unterstützung folgender Operationen von OpenCV

imgproc/ImageFiltering

/FA0210/ dilate

/FA0220/ erode

/FA0230/ morphologyEx

/FA0240/ Sobel

imgproc/Miscellaneous Image Transformations

/FA0250/ threshold

/FA0260/ adaptiveThreshold

/FA0270/ floodFill

feature2sd

/FA0280/ KeyPoint

/FA0290/ DMatch

4.2 API Optional

4.2.1 Allgemein

/FA0300/ Optionale Parameter für Einstellungen der Visualisierungen

/FA0400/ Lokale Auswahl Debug/Release-Modus

/FA0500/ Optionale nicht-blockierende Aufrufe für Streaming

4.2.2 Unterstützung folgender Operationen

/FA0600/ stitching

/FA0610/ ocl (OpenCL)

imgproc/Histograms

/FA0620/ calcHist

imgproc/Feature Detection

/FA0630/ Canny

/FA0640/ HoughCircles

4.3 GUI Pflicht

/FA0700/ Eine Visualisierung pro oben gelisteter Operation (siehe API Kriterien)

/FA0800/ Drei Visualisierungen für features2d/DMatch (drei der möglichen Wunsch-Visualisierungen, siehe unten)

/FA0900/ Zoomfunktion

4.4 GUI Optional

Vergleiche hierzu: ÜBERSICHTSEITE MIT TABS, OHNE VORSCHAUBILDER .

/FA1000/ Permanente GUI mit Historie

/FA1200/ Möglichkeit eine Filteroperation mit geänderten Parametern erneut anzuwenden

/FA1300/ Hohe Zoomstufen mit Zusatzinformationen (z.B. Pixelwerte), vgl. HOHE ZOOMSTUFEN MIT ZUSATZINFORMATIONEN

/FA1400/ Optionale Ausnutzung von mehreren Bildschirmen durch Fenstermodus

/FA1500/ Interaktive Überlagerung der Bilder durch Zusatzinformationen (Mouse over)

/FA1600/ Flexibler Umgang mit unterschiedlichen Bildschirm- und Bildauflösungen

/FA1700/ Suchleiste für alle Tabellen (z.B. jener der Übersichtsseite oder der Rohdatendatenanzeige)

/FA1710/ Spezielle Syntax zum Beispiel zur Gruppierung von Datensätzen

4.5 Mögliche Visualisierungen

4.5.1 Allgemein

/FA1800/ Darstellung von Rohdaten (vgl. ROHDATENANZEIGE)

/FA1810/ Abmessungen der Bilder

/FA1820/ Farbraum der Bilder (der in OpenCV genutzte Datentyp)

/FA1830/ Tabellarische Darstellung, z.B. der Matches, mit Filtermöglichkeit

/FA1840/ Diagramme (wie Histogramme)

/FA1900/ Darstellung der Bilder nebeneinander

4.5.2 Visualisierungen von Matches

/FA2000/ Basisvisualisierung Eine Visualisierung, die der 'drawMatches' Funktion von OpenCV ähnlich ist und unter BASISVISUALISIERUNG EINES MATCH-DATENSATZES skizziert wurde.

/FA2010/ Einzeichnen der Keypoints in die Bilder

/FA2020/ Verbinden der Matches durch Linien oder Pfeile

/FA2030/ Einfärben der Linien, Pfeile oder Punkte mit Falschfarben

/FA2040/ Ausblenden der Keypoints ohne Matches

/FA2050/ Auswahl von Matches anhand bestimmter Kriterien (z.B. via Histogramm)

/FA2060/ Manuelle Auswahl von Matches

/FA2070/ Automatische Zusammenfassung von Matches zu Gruppen

/FA2100/ **Projektionen** Eine Skizze findet sich hier: PROJEKTION .

/FA2110/ Einzeichnen von Linien / Formen

/FA2120/ Auswählen von zugehörigen Matches

/FA2130/ Die Linien / Formen werden auf das zweite Bild projiziert

/FA2140/ Automatische Gruppierung der Matches zu Flächen

/FA2200/ **Darstellung von Punkttranslationen von einem Bild zum anderen mit Pfeilen** Betrachte hierzu auch: DARSTELLUNG VON PUNKTTRANSLATIONEN .

/FA2210/ Pfeillänge und Richtung entsprechen der jeweiligen Translation

/FA2300/ **Stereoskopische Darstellung als Tiefenbild**

/FA2310/ Pixelfarbwerte entsprechen den jeweiligen Tiefenwerten

4.5.3 Visualisierungen für Filter

/FA2400/ Differenzbilder

/FA2500/ Überlagerungen

/FA2600/ Direkte Anwendung von Filtern auf ein oder zwei Bilder Beispiel: Anwendung eines Kantenfilters um die Auswirkungen z.B. einer Kantenglättung zu visualisieren

/FA2700/ Visualisierung über die Auswirkungen auf bestimmte Bildmetriken Beispiel: Überlagerung von Histogrammen beider Bilder oder Vergleich der Kontrastwerte von bestimmten Bildbereichen

5 Nichtfunktionale Anforderungen

5.1 Produktleistungen *Pflicht*

/NF0100/ Die GUI soll schnell starten und interaktiv bedienbar sein

/NF0200/ Möglichst kein Overhead im Release-Modus

5.2 Produktleistungen *Optional*

/NF0300/ Flexibler Umgang mit unterschiedlichen Bildschirm- und Bildauflösungen

/NF0400/ Integration in das OpenCV Test Framework

5.3 Qualitätsanforderungen *Pflicht*

- /NF0500/ Keine signifikanten Speicherlecks
- /NF0600/ Erweiterbarkeit um zusätzliche OpenCV Operationen und Visualisierungen
- /NF0700/ Modularer Aufbau (API und GUI)
- /NF0900/ Einhaltung der OpenCV und Qt Konventionen
- /NF1000/ Ausführliche Dokumentation der API und des GUI

5.4 Qualitätsanforderungen *Optional*

- /NF1100/ Dokumentation des internen Codes mit Werkzeug
- /NF1200/ OpenCV geeigneter Aufbau des Build-Systems
- /NF1300/ Abdeckung durch Tests
- /NF1400/ Keine Resource-Leaks
- /NF1500/ Threadsafety *Im C++11-Modus*
- /NF1600/ Toleranz gegenüber fehlerhaften API-Aufrufen
- /NF1700/ Kein undefiniertes Verhalten

6 Produktdaten

Rein aus der Konzeption unseres Projektes her, sind die meisten Produktdaten dem Typ der Dokumentation zu zurechnen.

/PD0100/ Handbuch

Es umfasst Erklärungen zur Benutzung der GUI, der einzelnen Visualisierungen und der Bibliothek. Des Weiteren sind ein Wegweiser zum Schnelleinstieg und Anleitungen zur Entwicklung von Erweiterungen des Projekts, insbesondere durch das Hinzufügen neuer Views, enthalten.

/PD0200/ API- und Codedokumentation

/PD0300/ FAQ

/PD0400/ Einstellungen

Einstellungen der Übersichtsseite und der einzelnen Visualisierungen, welche vom Benutzer verändert wurden, sollen gespeichert werden.

/PD0500/ Rohdaten

In vielen Views wird es möglich sein, die Rohdaten in einem gewünschten Format (z.B. CSV oder JSON) in einer Datei abzuspeichern.

/PD0600/ Ursprungsbilder sowie Visualisierungen in Form von Schnappschüssen

7 Systemmodell

- Das Interface wird aus öffentlichen Funktionen bestehen, die im Release-Modus nichts tun und im Debug-Modus, wie gewünscht, die nicht-öffentlichen Debug-Methoden aufrufen.
- Die nichtöffentlichen Debug-Funktionen erzeugen eine graphische Visualisierung und fügen diese dem Thread-lokalen Hauptfenster hinzu.
- Die Verwaltung des Hauptfensters wird vermutlich über einen Singleton stattfinden.
- Die Visualisierungen werden sich, wie in der modernen Softwareentwicklung üblich, aus modularen Komponenten zusammensetzen.
- Die Bibliothek selbst wird über die Funktionen des öffentlichen Userinterfaces angesteuert die lediglich prüfen ob der Debugmodus aktiv ist und gegebenenfalls die eigentlichen Funktionen im nichtöffentlichen, technischen Interface aufrufen.

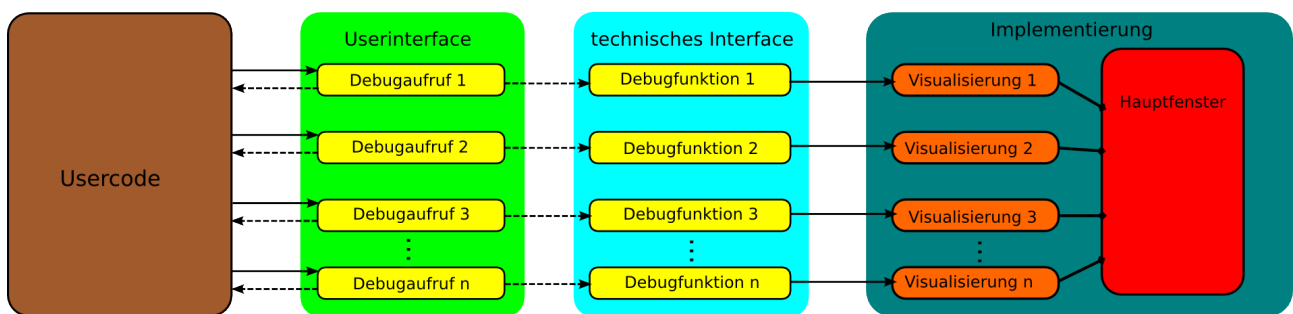


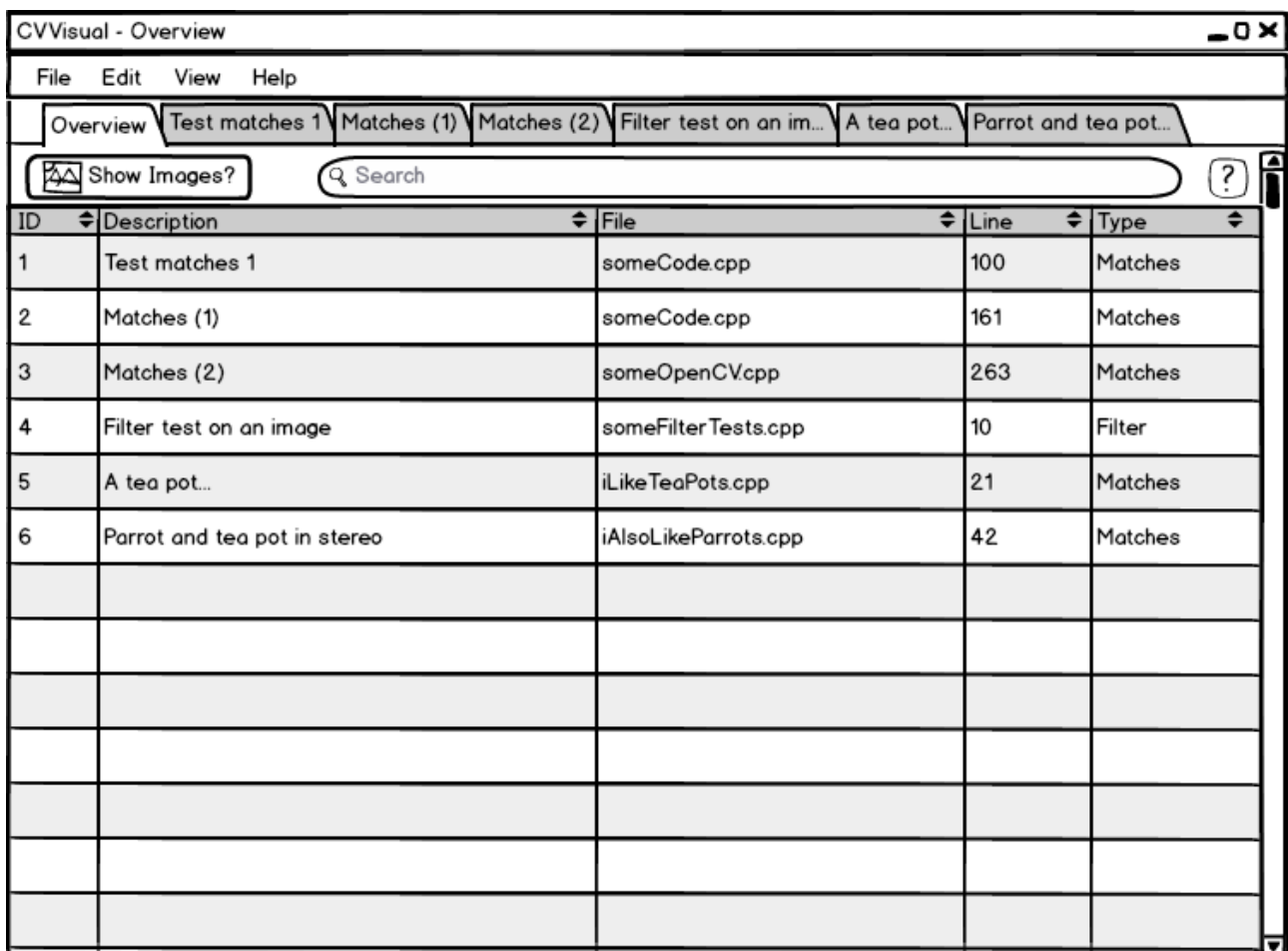
Abbildung 1: Das Modul besteht aus drei Layern bei denen jeweils die Äußeren die Inneren aufrufen

8 Bedienoberfläche

- Die Bedienoberfläche wird in Qt implementiert sein.
- Sie wird aus einem Hauptfenster pro Thread bestehen.
- Die verschiedenen Debug-Ansichten werden entweder im Hauptfenster oder in einzelnen Fenstern dargestellt.
- Daten von vorherigen API-Aufrufen werden auf einer Übersichtsseite tabellarisch dargestellt

8.1 Skizzen

8.1.1 Übersichtseite mit Tabs, ohne Vorschaubilder



The screenshot shows a Qt-style window titled "CVVisual - Overview". It has a menu bar with "File", "Edit", "View", and "Help". Below the menu bar is a tab bar with tabs: "Overview", "Test matches 1", "Matches (1)", "Matches (2)", "Filter test on an im...", "A tea pot...", and "Parrot and tea pot...". Below the tabs is a toolbar with a button labeled "Show Images?" and a search bar labeled "Search". Below the toolbar is a table with the following data:

ID	Description	File	Line	Type
1	Test matches 1	someCode.cpp	100	Matches
2	Matches (1)	someCode.cpp	161	Matches
3	Matches (2)	someOpenCV.cpp	263	Matches
4	Filter test on an image	someFilterTests.cpp	10	Filter
5	A tea pot...	iLikeTeaPots.cpp	21	Matches
6	Parrot and tea pot in stereo	iAlsoLikeParrots.cpp	42	Matches

Abbildung 2: Übersichtseite mit Tabs, ohne Vorschaubilder

Der Benutzer kann die einzelnen Datensätze auswählen und mithilfe der Suchleiste in der Tabelle zu suchen. Wenn er auf einen Datensatz in der Tabelle drückt, öffnet er die Visualisierung für diesen Datensatz. Außerdem kann er mit dem Knopf **Show Images** Vorschaubilder in der Tabelle anzeigen lassen (nächste Skizze) und über das Menü **View** einstellen, ob er die Visualisierungen als Tabs im Hauptfenster oder als eigene Fenster angezeigt haben möchte.

8.1.2 Übersichtseite mit Tabs und Vorschaubildern



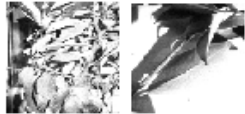
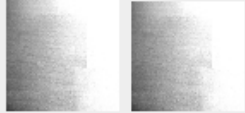



CVVisual - Overview						
File Edit View Help						
Overview Test matches 1 Matches (1) Matches (2) Filter test on an im... A tea pot... Parrot and tea pot...						
 Show Images? <input type="text" value="Search"/> ?						
ID	Images	Description	File	Line	Type	
1		Test matches 1	someCode.cpp	100	Matches	
2		Matches (1)	someCode.cpp	161	Matches	
3		Matches (2)	someOpenCV.cpp	263	Matches	
4		Filter test on an image	someFilterTests.cpp	10	Filter	
5		A tea pot...	iLikeTeaPots.cpp	21	Matches	
6		Parrot and tea pot in stereo	iAlsoLikeParrots.cpp	42	Matches	

Abbildung 3: Übersichtseite mit Tabs und Vorschaubildern

Das gleiche Übersichtsfenster, nur dass in der Tabelle Vorschaubilder angezeigt werden. Durch Überfahren jener mit der Maus, werden größere Bilddarstellungen eingeblendet.

8.1.3 Übersichtseite mit Fenstern und Vorschaubildern

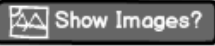


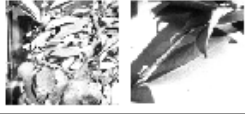




CVVisual - Overview					
File Edit View Help					
 <input type="text" value="Search"/> 					
ID	Images	Description	File	Line	Type
1		Test matches 1	someCode.cpp	100	Matches
2		Matches (1)	someCode.cpp	161	Matches
3		Matches (2)	someOpenCV.cpp	263	Matches
4		Filter test on an image	someFilterTests.cpp	10	Filter
5		A tea pot...	iLikeTeaPots.cpp	21	Matches
6		Parrot and tea pot in stereo	iAlsoLikeParrots.cpp	42	Matches

Abbildung 4: Übersichtseite mit Fenstern und Vorschaubildern

Übersichtseite mit Fenstern statt Tabs und Vorschaubildern. Dies ermöglicht dem Benutzer mehrere Visualisierungen nebeneinander anzusehen - und damit auch mehrere Bildschirme zu nutzen.

8.1.4 Basisvisualisierung eines Match-Datensatzes

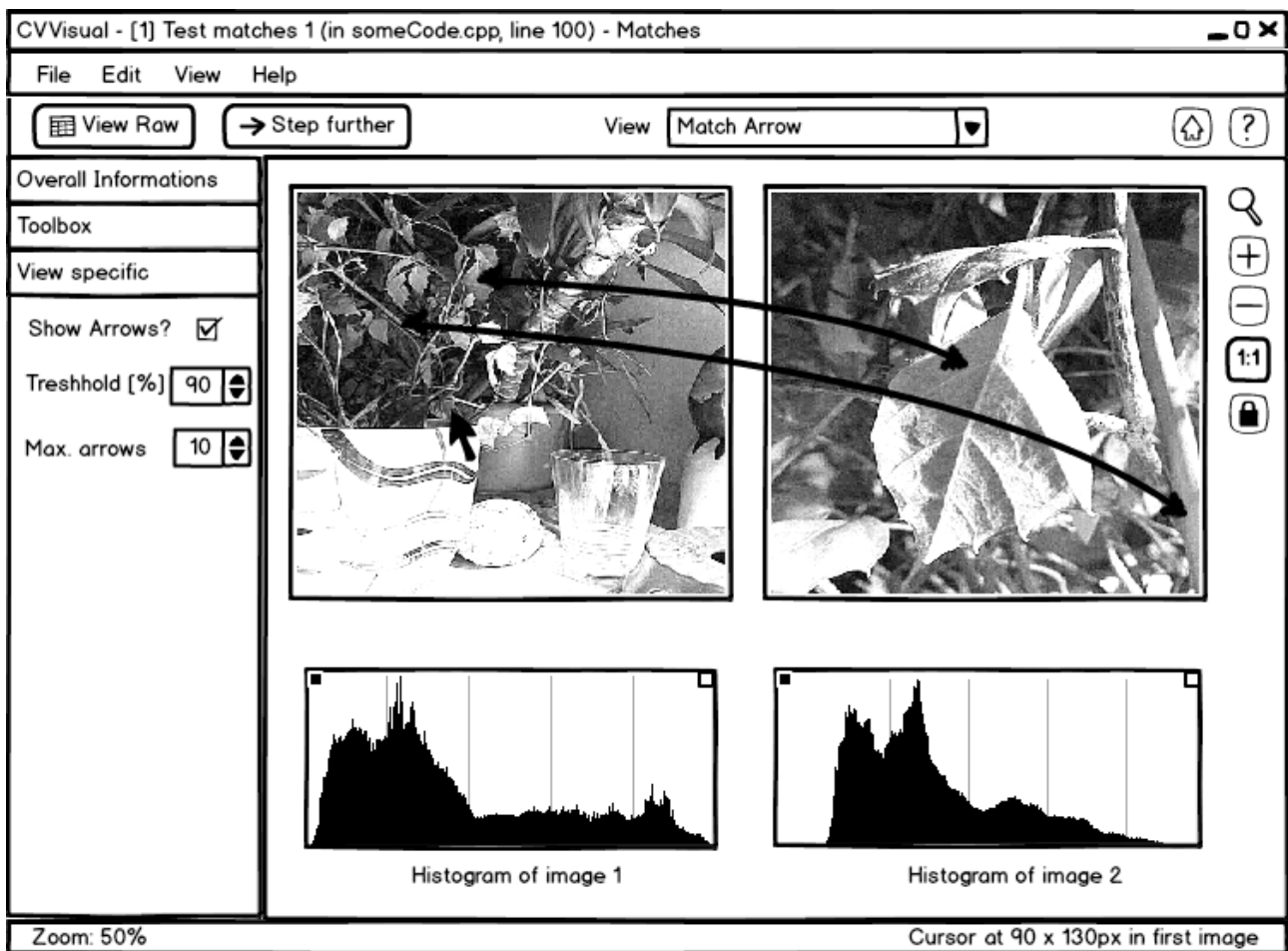


Abbildung 5: Basisvisualisierung eines Match-Datensatzes

Unten in der Fensterleiste findet sich der aktuelle Zoomfaktor und die aktuelle Position in einem der beiden Bilder. Wichtig: Die Position entspricht jener im gesamten Bild - nicht jener im angezeigten Ausschnitt.

Mit dem Button **View Raw** kann der Benutzer die Rohdatenanzeige für den aktuellen Datensatz anzeigen. Der Button **Step further** wird angezeigt, sofern die zu debuggende Anwendung wegen dieser Visualisierung blockiert. Wobei damit die Visualisierung nicht geschlossen wird.

Die View-Auswahl und die Zoom-Knöpfe am rechten Rand sind selbsterklärend. Bei Eindrücken des Schlossknopfes versucht das Programm die Anzeigen beider Bilder zu synchronisieren.

8.1.5 Rohdatenanzeige

CVVisual - [1] Test matches 1 (in someCode.cpp, line 100) - Raw matches

File Edit View Help

Return to match view Search

ID 1	ID 2	x1	y1	x2	y2	angle 1 [°]	size 1	angle 2 [°]	size 2	response 1	response 2	distance
1	2	58	78	241	190	30	2	20	5	0.4	0.5	0.1
2	1	103	50	114	101	45	1	42	3	0.1	0.6	0.9

Matches shown: 2 Belongs to: [1] Test matches 1 (in someCode.cpp, line 100)

Abbildung 6: Rohdatenanzeige

Tabellarische Anzeige der Rohdaten, nach dem der Benutzer in der vorherigen Skizze auf den Knopf  gedrückt hat.

Dem Benutzer ist es wieder möglich mit Hilfe der Suchleiste in den Daten zu suchen und mit gedrückter *STRG*-Taste auch mehrere Einträge zu markieren. Über das Kontextmenu ist es des Weiteren möglich entweder die ausgewählten Einträge in der Visualisierung hervorzuheben oder sie als *JSON* oder *CSV* in die Zwischenablage zu kopieren.

8.1.6 Hohe Zoomstufen mit Zusatzinformationen

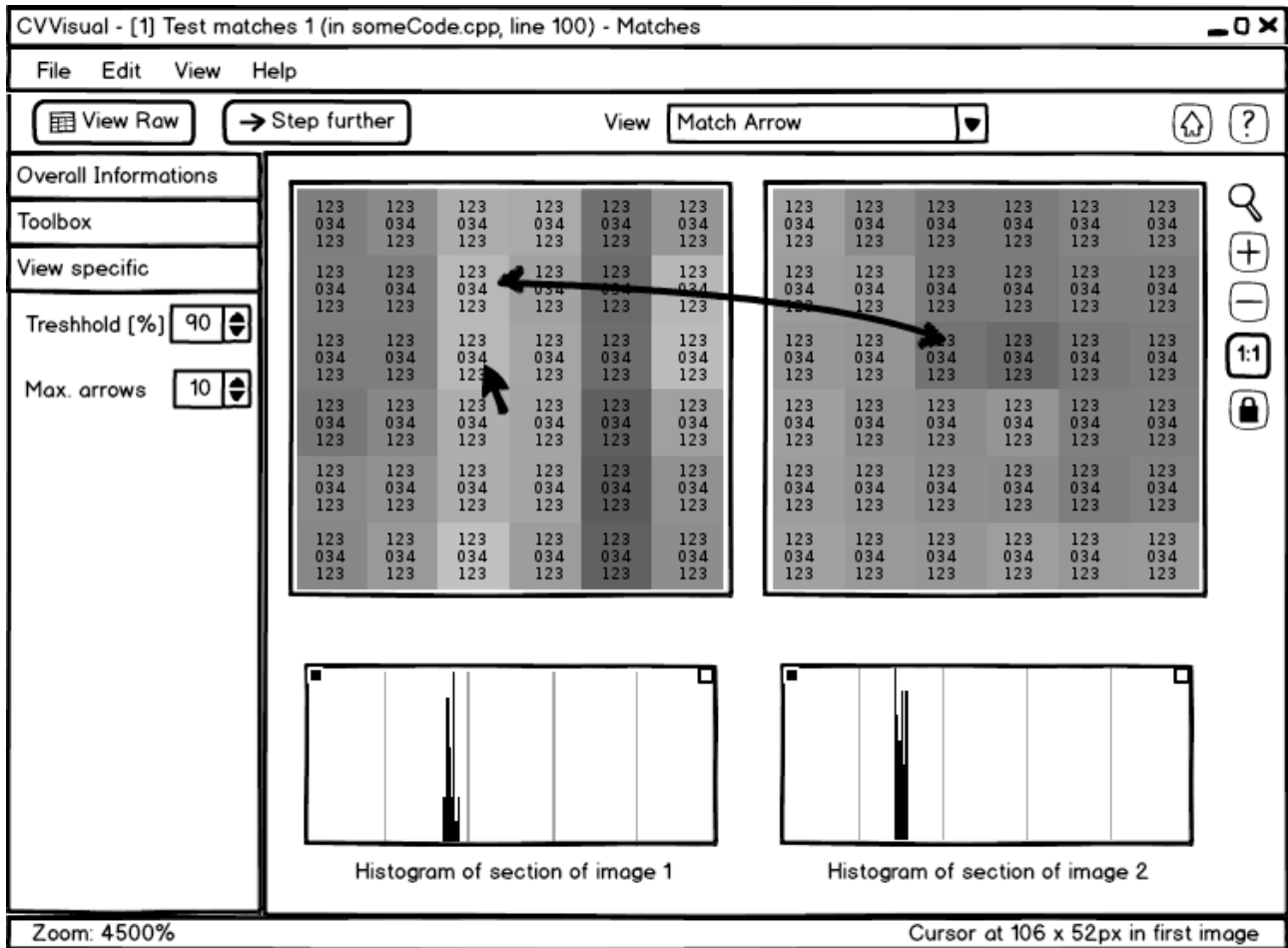


Abbildung 7: Hohe Zoomstufen mit Zusatzinformationen

Der Benutzer sieht bei einem hohen Zoomfaktor Zusatzinformationen im Bild, in diesem Fall die Werte für Rot, Grün, Blau für jedes einzelne Pixel.

8.1.7 Projektion

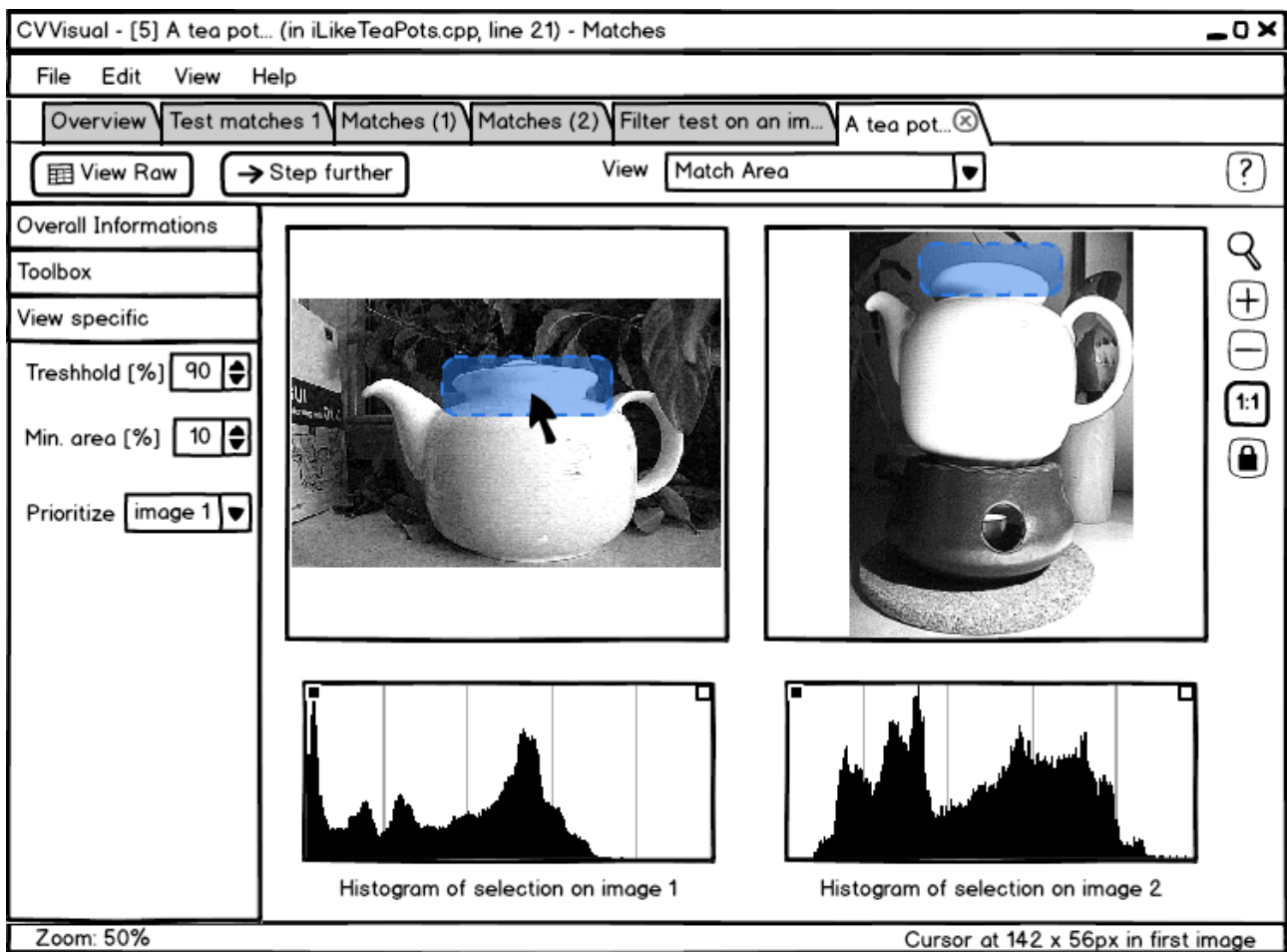


Abbildung 8: Projektion

Projektion einer automatischen Gruppierung von Matches von einem auf das andere Bild. Hierbei wird jeweils das Histogramm des vom Benutzer ausgewählten Bereichs angezeigt.

8.1.8 Darstellung von Punkttranslationen

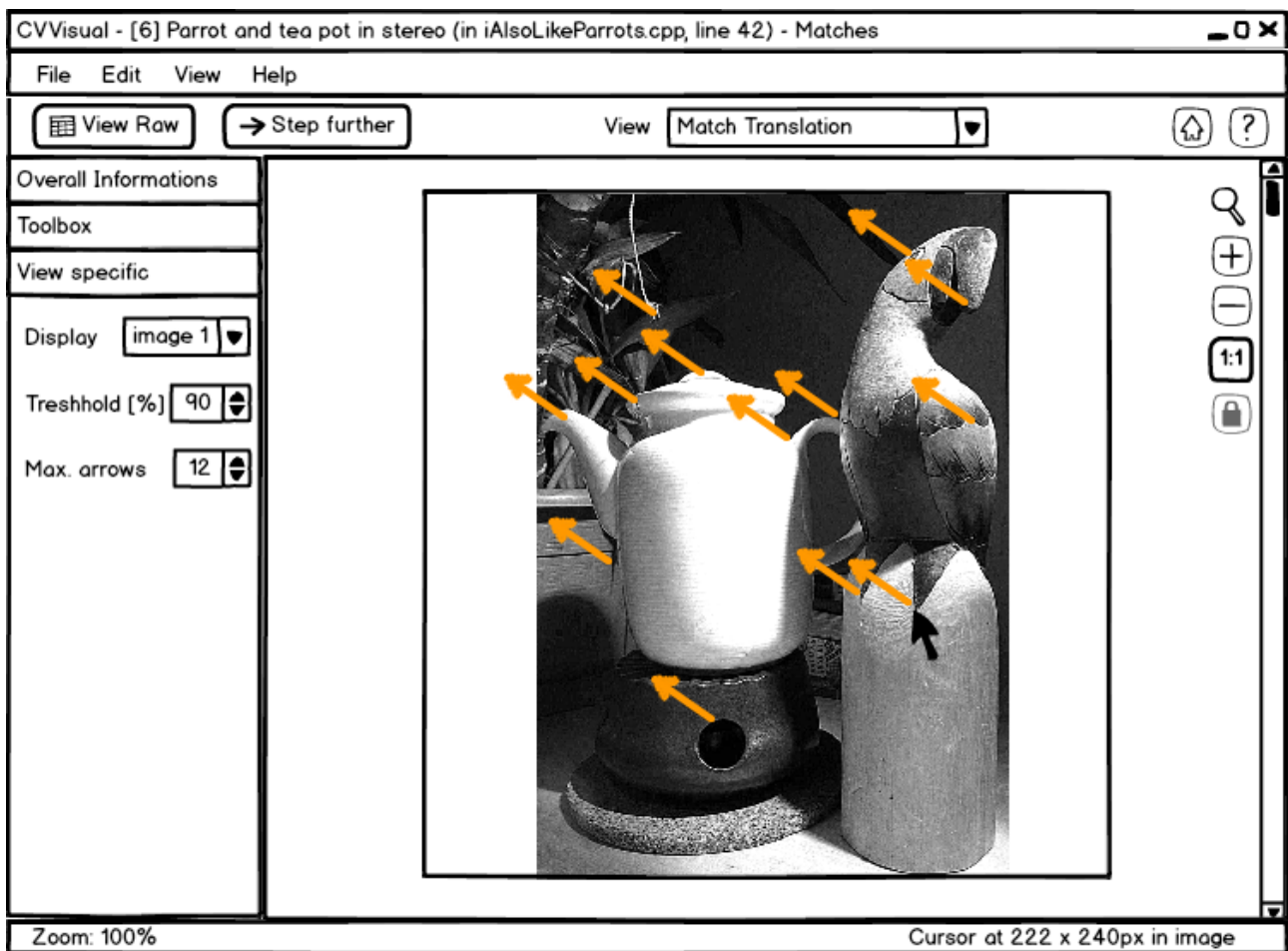


Abbildung 9: Darstellung von Punkttranslationen

Darstellung von Punkttranslationen von einem Bild zum anderen mit Pfeilen. Die Länge und Richtung des jeweiligen Pfeils entsprechen der Translation oder Verschiebung des Startpunktes im anderen Bild.

9 Testfälle und Testszenarien

Unser Projekt ist eine reine Debug-Bibliothek, die es dem Benutzer möglichst einfach macht, seine Bilddaten zu visualisieren. Deshalb enthält der Funktionsumfang im Grunde genommen nur Visualisierungen und die meisten Testszenarios besitzen auf Grund dessen die folgende Struktur:

1. Der Programmierer schreibt den gewünschten API-Aufruf an die gewünschte Stelle in seinen Quelltext.
2. Ein Fenster öffnet sich und visualisiert die beim Aufruf übergebenen Daten.
3. Nun kann er mit der Visualisierung arbeiten, sie in den Einstellungen anpassen oder die Visualisierung wechseln.
4. Bei blockierenden Aufrufen klickt er nun auf einen Button, der das aufrufende Programm weiterlaufen lässt.
5. Er sucht entweder weiter Fehler in seinem Programm, oder sieht die angefallenen Datensätze in der Übersichtsseite durch und visualisiert die gewünschten.
6. Die Debugumgebung lässt sich ohne Fehler beenden.

9.1 Tests für Filterview

/TF010/ Anzeige des Bildes/der Bilder

/TF020/ Auswählen und Wechseln der Filter

/TF030/ Ändern der Parameter eines Filters

/TF040/ Bedienung bei einem großen Bild und geringer Rechenleistung

/TF050/ Benutzung mit mehr als einem Bild

9.2 Tests für Matchview

/TF060/ Anzeige des Bildes/der Bilder

/TF070/ Auswählen und Wechseln von Views


/TF080/ Sämtliche Buttons und Schieberegler testen

/TF090/ Bedingung mit großen Bildern und mit geringer Rechenleistung

9.3 Beispielhafter Ablauf

Der Benutzer tippt irgendwo in seinem Code die folgende Zeile, wobei `inputImg1` und `inputImg2` zwei Bilder und `matches` die von OpenCV gefundenen Matches sind.

```
cvvisual::showMatches(inputImg1, inputImg2, matches);
```

Dieser Code erzeugt ein Fenster, mit dem Standardview für Matches. Der View zeigt die Bilder nebeneinander an und zeichnet zwischen den einzelnen Matches Pfeile (siehe Basisvisualisierung eines Match-Datensatzes). Der Benutzer hat nun die Auswahl sich zusätzliche Informationen anzeigen zu lassen, wie den *Matchwert* der einzelnen Matches durch Einfärben der Pfeile in Falschfarben anzeigen zulassen, oder zu einem anderen View mit anderen Optionen zu wechseln. In dem anderen View werden dieselben Bilder benutzt. Jetzt drückt er den  Knopf um sich die gesamten Rohdaten (wie beispielsweise die Punkt-Koordinaten der einzelnen Matches) im nun aufgehenden Fenster anzeigen

zu lassen (siehe Rohdatenanzeige). Für eine angenehmere Bedingung steht ihm eine Suchleiste mit einigen Befehlen zur Verfügung. Mit ihrer Hilfe sortiert er die Liste nach der x-Koordinate des ersten Punktes jedes Matches. Er wählt die erste Zeile der Tabelle durch Rechtsklicken aus und exportiert mit dem erscheinenden Kontextmenu den Zeileninhalt - ein Match - in die Zwischenablage als JSON. Dieses JSON schreibt er für später in die Ursprungsdatei. Nun kehrt der Benutzer durch Klicken des **Return to match view** Knopfes zur Matchview zurück und zoomt dort etwas im Bild herum und untersucht für ihn wichtige Bildteile. Nachdem der Benutzer, das Gewünschte gesehen hat drückt er den **Step further** Knopf, damit sein Programm weiter läuft. Wenn sein Programm sich beendet, beendet sich auch CVVisual und schließt alle Fenster.

10 Abgrenzungskriterien

Unser Projekt grenzt sich durch existentes Design gegenüber „random code“ ab. Darüber hinaus ist uns keine andere OpenSource-Lösung bekannt, die mit unserer vergleichbare Ziele verfolgt.

Wichtig: Unser Projekt ist kein Stand-Alone-Programm und wird voraussichtlich keinen rückläufigen Datenfluss unterstützen.

11 Entwicklungsumgebung

11.1 Quellcode

1. GCC 4.8.0 oder neuer
2. Qt 5
3. C++11
4. OpenCV 2.4.6 oder neuer
5. GNU/Linux (im besonderen Ubuntu 12.04)
6. CMake 8.2.7 oder neuer

11.2 Dokumentation

Zur Dokumentation werden im Wesentlichen die auch von OpenCV verwendeten Werkzeuge benutzt. Die folgende Aufzählung wird im Verlauf der Projektentwicklung unter Umständen noch erweitert.

SPHINX Ein in Python geschriebenes Werkzeug, mit dessen Hilfe das Handbuch und das FAQ geschrieben wird.

Doxygen Ein weit verbreitetes Werkzeug um aus den Kommentaren im Quelltext unseres Projektes eine übersichtliche Quelltextdokumentation zu erzeugen (vgl. Produktdaten). Es wird in Verbindung mit SPHINX verwendet.

12 Lizenz

Unser Projekt ist OpenSource, weshalb jegliche Dokumentation und Quellcode unter freien Lizenzen stehen. Im folgenden werden nun kurz die Lizenzen für Dokumentation und Quelltext erläutert.

12.1 Quellcode

Der Quelltext dieses Projektes steht in seiner Gesamtheit unter der BSD Clause 3 Lizenz, da auch OpenCV unter dieser Lizenz steht. Die BSD Clause 3 Lizenz erlaubt es anderen Entwicklern den Quelltext unseres Projektes, wie auch dessen Binärformen, weiterzuverteilen und zu verändern, solange der folgende Lizenzabschnitt im Programm selbst, dem Quelltext und der Dokumentation enthalten ist. Außerdem ist es verboten, den Namen CVVisual oder die Namen der Mitentwickler zur Produktwerbung zu verwenden, ohne dass eine vorherige Erlaubnis eingeholt wurde.

Copyright (c) 2013, CVVisual PSE Team
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the cvvisual project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE CVVISUAL DEVELOPERS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Diese Lizenz hat außerdem, gegenüber z.B. der GNU GPL, den Vorteil, dass es auch eine Verwendung unseres Projektes im Umfeld proprietärer Lizenzen erlaubt, und damit hoffentlich auch von Unternehmen genutzt werden kann.

12.2 Dokumentation

Die Dokumentation, mit samt aller Bilder und Texte, steht unter der CC-BY-SA Lizenz. Sie erlaubt es anderen die Dokumentation, zu kopieren, zu verändern und zu kommerziellen Zwecken zu verwenden. Wobei hierbei immer der Autor genannt werden muss. Wie auch abgeleitete Werke unter derselben, oder einer gleichwertigen, Lizenz gestellt werden müssen.

13 Glossar

13.1 Allgemein

API Application Programming Interface. Eine Schnittstelle (s. Interface), über welche andere Programme auf der Quelltextebene auf dahinter verborgene Funktionalität zugreifen können

Augmented Reality „Erweiterung der Realität“ durch einen Computer, etwa bei der Einblendung von Informationen in ein Bild der Umgebung, das auf einem Smartphone angezeigt wird.

Bug Fehlerhaftes Verhalten eines Programmes.

Binärform Hier das Programm in für den Computer direkt verwendbarer Form im Gegensatz zum Quellcode. Anders als dort ist hier kaum sichtbar, wie das Programm genau arbeitet, weshalb bei OpenSource-Projekten gerade der Quelltext offen liegt (vgl. OpenSource).

Datenstrom Daten, die beispielsweise während der Ausführung eines Programms fließen, wobei das Ende dieses Flusses nicht absehbar ist.

Debug-Modus Modus, bei dem zusätzliche Informationen angezeigt werden, um dem Programmierer das Auffinden und Beheben von Bugs (kurz *Debugging* oder *Debuggen*), hier insbesondere Programmierfehlern, zu erleichtern. Vgl. Release-Modus.

Debug-Visualisierung Hier eine Visualisierung, die den Benutzer beim Debuggen unterstützt, indem sie relevante Daten zu den übergebenen Bildern anzeigt und diese damit leicht verständlich darstellt.

FAQ Engl. für „Häufig gestellte Fragen“ enthält sie viele Fragen, die besonders neue Benutzer sich stellen, wenn sie anfangen ein Projekt zu nutzen.

Falschfarben Verwendung von Farben in einem Bild, die sich von den natürlichen, erwarteten Farben stark unterscheiden, um etwa Details hervorzuheben.

Filter In der Bildverarbeitung die Veränderung eines Bildbereiches mithilfe eines bestimmten Algorithmus.

freie Lizenz Eine Lizenz, die im Sinne der Open Source Initiative, die Veränderung und Weitergabe des unter ihr lizenzierten Inhalts ermöglicht.

GNU "GNU is Not Unix". Freies Betriebssystem und Software.

GNU GPL *General Public License*. Eine freie Software-Lizenz. Alle auf einer unter dieser Lizenz stehenden Software aufbauende Software steht ebenfalls unter dieser Lizenz.

GNU/Linux Das GNU-Betriebssystem in Kombination mit einem Linux-Kern

GNU-Projekt Das Projekt zur Erstellung von GNU-Betriebssystem und Software.

GUI Graphical User Interface, zu deutsch Graphische Benutzeroberfläche. Stellt Funktionen graphisch dar, sodass der Benutzer beispielsweise per Mausklick damit interagieren kann; im Gegensatz zu textbasierten Benutzerschnittstellen (vgl. Interface).

Interface *Schnittstelle*. Ein Zugriffspunkt auf Funktionalitäten von außen, der einerseits den Zugriff erleichtern und andererseits die eigentliche Funktionalität verbergen soll.

Kompilieren Umwandlung eines Quellcodes in eine für den Computer verständliche Form, mithilfe eines Compilers genannten Programms.

Matches Durch OpenCV erzeugte Verknüpfungen zwischen zwei Bildbereichen bzw. Bildpixeln, welche vom Benutzer an die API übergeben werden.

- Mouse over** Information über das Element einer GUI, auf dem der Mauszeiger ruht, wird angezeigt.
- Parallelrechner** Rechner, in dem mehrere Threads gleichzeitig nebeneinander ausgeführt werden können (vgl. Thread).
- OpenCV Test Framework** Stellt Funktionen zum Testen zur Verfügung, etwa Überprüfungen, ob zwei Matrizen gleich sind.
- Open Source** Software, bei welcher der Quellcode frei zugänglich gemacht wird. Dies erlaubt unter anderem die Weiterverwendung und -entwicklung durch andere.
- Overhead** Zusätzlicher Speicher- oder Zeitaufwand.
- proprietär** In diesem Zusammenhang Software, die nicht unter einer freien Lizenz steht.
- Quellcode oder Quelltext** In einer Programmiersprache geschriebener Text, der zu einem Computerprogramm umgewandelt werden kann (vgl. auch Binärform).
- "random code"** Individueller Wegwerfcode der nur für eine einzige Probleminstanz hilfreich ist und sich meist durch nicht existentes Design auszeichnet.
- Release-Modus** Veröffentlichungs-Modus, in dem keine zusätzlichen Debug-Informationen angezeigt werden, also der Modus, in dem das fertige Programm läuft.
- Resource-Leak** Das Auftreten der Situation, dass Ressourcen irgendeiner Art (Speicher, Dateien, usw.) zwar alloziert werden, aber nach Verwendung nicht mehr an das System zurückgegeben werden.
- Rohdaten** Daten, welche direkt und ohne wirkliche Aufarbeitung, aus den vom Entwickler beim API-Aufruf übergebenen Datenstrukturen stammen.
- Speicherleck** *engl. memory leak* Vgl. Resource-Leak; die Ressource ist in diesem Fall Speicher.
- Stand-Alone-Programm** Programm, das für sich alleine funktioniert.
- Streaming** Hier das Weiterlaufen des Datenstroms.
- Tab** Hier ein registerkartenähnlicher Teil einer GUI.
- Thread** Ausführungsstrang in einem Programm. Der Begriff wird insbesondere im Zusammenhang mit Mehrfachkernsystemen, wo mehrere Threads gleichzeitig nebeneinander laufen können, oft verwendet.
- thread-lokal** Auf einen Thread beschränkt (vgl. Thread).
- Threadsafety** Es ist sichergestellt, dass mehrere Threads sich nicht gegenseitig stören, etwa beim Speichern von Daten.
- Translation Unit** Eine Einheit, die einzeln im Ganzen kompiliert wird (s. Kompilieren); ein Projekt teilt sich meist in mehrere solcher Einheiten auf.
- Undefiniertes Verhalten** Befehle, deren Verwendung dazu führt, dass der C++-Standard *keinerlei* Verhaltensgarantien irgendeiner Art für das gesamte Programm mehr gibt. Etwas Umgangssprachlich: Der Standard untersagt die Verwendung.
- View** Zusammengehörige Visualisierungen eines bestimmten OpenCV-Features (oder einer Featureart).

13.2 OpenCV

Weiterführende Informationen sind auf docs.opencv.org zu finden.

adaptiveThreshold OpenCV-Methode, die mittels eines adaptiven Threshold (s. unten) Graustufenbilder in (u.U. invertierte) Binärbilder umwandeln kann.

calcHist Berechnet ein Histogramm.

Canny Kantenerkennung (mithilfe des Canny86-Algorithmus).

Dilatation Berechnung des Bereiches in einem Bild, der von einer Maske abgedeckt wird, wenn sich deren Bezugspunkt (oft der Mittelpunkt) durch den ganzen zu untersuchenden Bildbereich bewegt.

DMatch Klasse für das Matching (vgl. Matches).

Erosion Prüft, ob eine Maske, etwa eine geometrische Figur, vollständig in ein Bild bzw. einen Bildbereich passt und gibt u.U. ein Bild zurück, in dem nur die überdeckten Teile erhalten sind. Bildet zusammen mit der Dilatation zwei der grundlegenden Bildverarbeitungsmethoden.

floodFill Bei dieser Methode wird ein zusammenhängender Bereich des Bildes mit der übergebenen Farbe ausgefüllt.

HoughCircles Findet Kreise in einem Graustufenbild (unter Benutzung der Hough-Transformation).

KeyPoint Klasse, die Daten eines Punktes (Koordinaten, Nachbarschaftsgröße etc.) enthält.

morphologyEx diese Methode von OpenCV erlaubt fortgeschrittene morphologische Transformationen unter Benutzung und mehrfacher Anwendung von Dilatation und Erosion.

ocl Das OCL-Modul stellt Implementierungen von OpenCV-Funktionalität für Geräte, welche die Open Computing Language (kurz OpenCL), ein Interface über Parallelrechner, benutzen, zur Verfügung.

Sobel-Operator Ein Kantenerkennungs-Filter.

Stitching Zusammenfügen mehrerer Bilder mit zusammenhängenden Bereichen an den Rändern zu einem großen Bild, etwa von Einzelfotografien zu einem Panorama.

threshold Diese Methode eröffnet verschiedene Möglichkeiten, die Elemente eines Arrays auf ein bestimmtes Niveau zu trimmen, auf binäre Werte herunterzubrechen und Ähnliches.

14 Literatur

Designed for Use – Create Usable Interfaces for Applications and the Web / Lukas Mathis Die vorliegenden GUI-Entwürfe sind stark von diesem Buch beeinflusst, da es wichtige Techniken, Methoden und Denkmuster der UI-Entwicklung vermittelt. *Und das in einer für Informatiker verständlichen Sprache.*

OpenCV Dokumentation Eine wichtige Quelle aus offensichtlichen Gründen.

Technisches Schreiben – (nicht nur) für Informatiker / Peter Rechenberg Ein anschaulich und verständlich geschriebenes Buch, das hilft den eigenen Schreibstil zu verbessern. Und damit die Verständlichkeit der geschriebenen Dokumentation (vgl. Produktdaten) zu erhöhen.