

# CVVisual

Ein Debug-Framework für OpenCV

Andreas    Clara    Erich    Florian    Johannes    Nikolai  
               Raphael

20. Juni 2014

# Gliederung

- Einführung in OpenCV
- Motivation
- Anwenderfeatures
- Gui-Demo
- Dokumentation
- Architektur
- API
- Ausblick

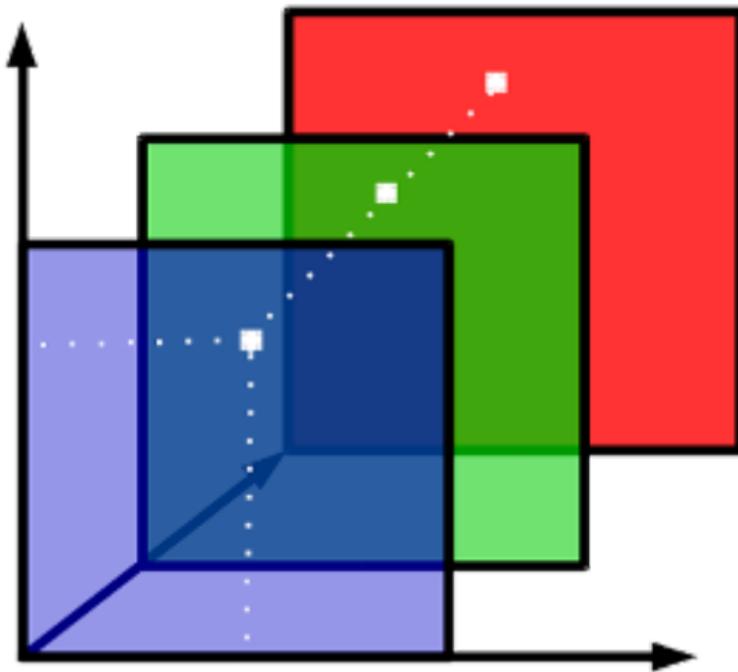
# Einführung in OpenCV

# Überblick

- Bildverarbeitung
- weite Verbreitung
- Matrizen als Grundlage
- Filter + Matches

# Matrizen

Bild = mehrdimensionale Matrix



# Filter

Berechnung auf Umgebung jedes Pixels

5	7	3	5	5	5
3	2	6	7	6	5
2	3	2	4	6	6
3	3	5	6	4	5
1	4	6	2	2	4
3	4	7	5	6	5

# Filter

Beispiel dilate: helle Flächen werden größer



# Filter

Beispiel dilate: helle Flächen werden größer



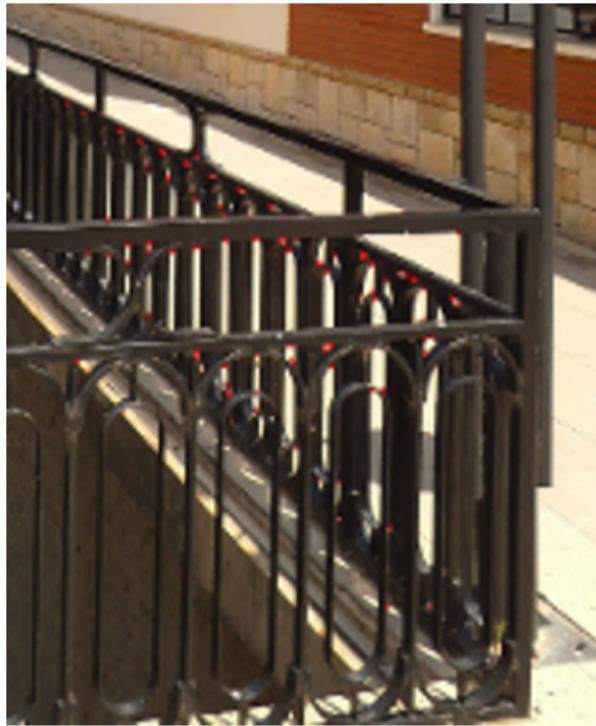
# Filter

Beispiel dilate: helle Flächen werden größer



## Matches

Keypoints = charakteristische Punkte



# Matches

Match = Paar aus Keypoints



# Motivation

# Debuggen von OpenCV

Systematisches Debugging statt „Random Code“

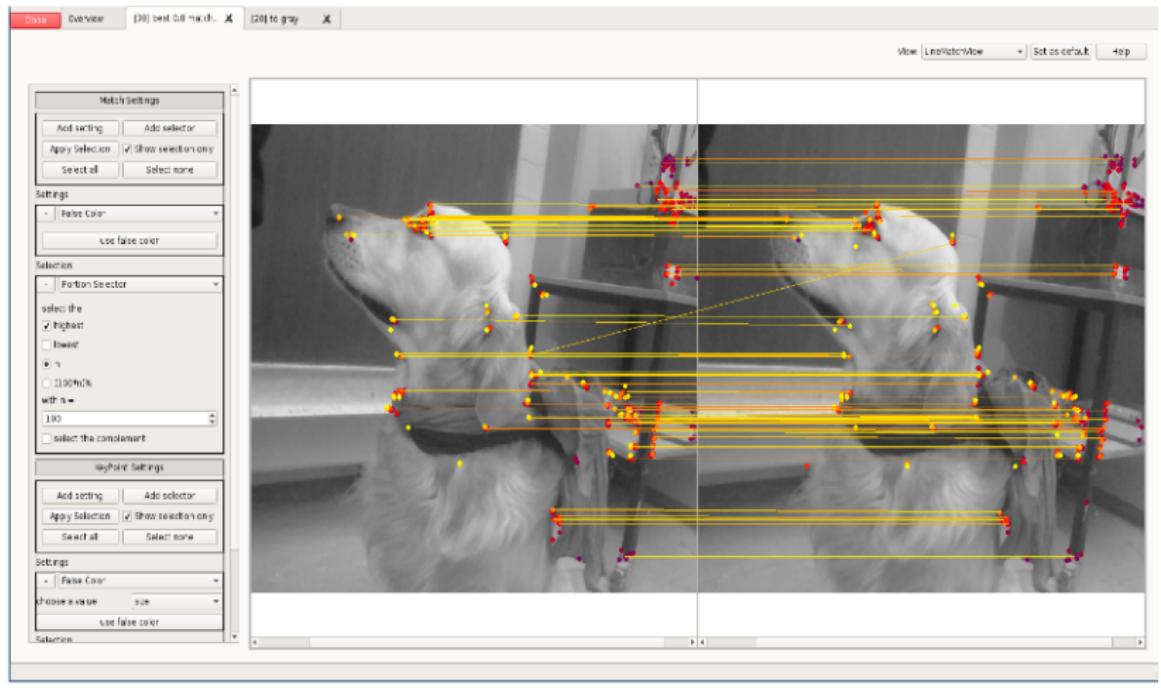
```
#ifdef DEBUG
    Mat img_matches;
    drawMatches( img_1, keypoints_1, img_2, keypoints_2,
                 good_matches, img_matches, Scalar::all(-1), Scalar::all(-1),
                 vector<char>(), DrawMatchesFlags::NOT_DRAW_SINGLE_POINTS );
    imshow("good matches", img_matches);
#endif
```

*versus*

```
cv::debugMatches(img1, img2, keypoints_1, keypoints_2, good_matches);
```

# Ziele

## Visualisierung von Matritzen, Filtereffekten und Matches



# Anwenderfeatures

# Verwendung

```
std::string imgIdString = "imgRead" + toString(imgId);
cvv::showImage(imgRead, CVVISUAL_LOCATION, imgIdString);

// convert to grayscale:
cv::Mat imgGray;
cv::cvtColor(imgRead, imgGray, CV_BGR2GRAY);
cvv::debugFilter(imgRead, imgGray, CVVISUAL_LOCATION,
                 "to gray", "SingleFilterView");
```

# Übersicht

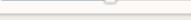
## Übersicht über alle Aufrufe

CVVisual | main window

Close Overview Help

No grouping specified, use #group to specify one

ID	Image 1	Image 2	Description	Function	File	Line	Type
1			IMG_1353.JPG	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	22	singleImage
2			IMG_1130.JPG	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	22	singleImage
3			IMG_1396.JPG	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	22	singleImage
4			IMG_1397.JPG	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	22	singleImage

Zoom 

☰ 🔍 ⌂ ⌂ ⌂

# Übersicht

## Filterbar

CVVisual | main window

**#type match**

ID	Image 1	Image 2	Description	Function	File	Line	Type
19			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match
20			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match
21			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match
22			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match

Zoom    

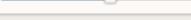
# Übersicht

## Sortierbar

CVVisual | main window

**#sort by line desc**

ID	Image 1	Image 2	Description	Function	File	Line	Type
19			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match
20			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match
21			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match
22			<no description>	int main(int, char**)	/home/parttimenerd/Studium/PSE/cvvisual_test/main.cpp	59	match

Zoom 

# Übersicht

## Gruppierbar

CVVisual | main window

Close Overview #group by description Help

5		IMG_1454.JPG	int main(int, char**)	/home/partimenerd/Studium/PSE/cvisual_test/main.cpp	22	singleImage
---	--	--------------	-----------------------	-----------------------------------------------------	----	-------------

IMG\_1455.JPG

ID	Image 1	Description	Function	File	Line	Type
6		IMG_1455.JPG	int main(int, char**)	/home/partimenerd/Studium/PSE/cvisual_test/main.cpp	22	singleImage

erode

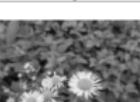
ID	Image 1	Image 2	Description	Function	File	Line	Type
7			erode	int main(int, char**)	/home/partimenerd/Studium/PSE/cvisual_test/main.cpp	36	filter

Zoom

## Übersicht

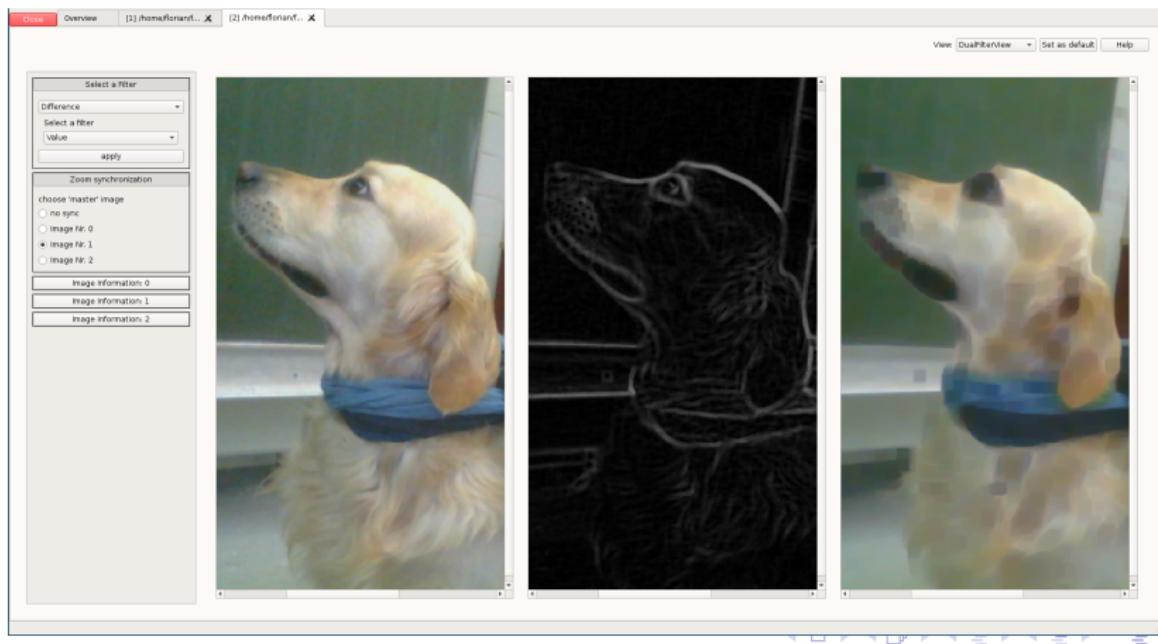
CVVisual | main window

**Close**   **Overview**   **Help**

<no description>						
ID	Image 1	Image 2	Description	Function	File	Line Type
19			<no description>	int main(int, char**)	/home/partimenerd/Studium/PSE/cvvisual_test/main.cpp	59 match
20			<no description>	int main(int, char**)	/home/partimenerd/Studium/PSE/cvvisual_test/main.cpp	59 match
21			<no description>	int main(int, char**)	/home/partimenerd/Studium/PSE/cvvisual_test/main.cpp	59 match
22			<no description>	int main(int, char**)	/home/partimenerd/Studium/PSE/cvvisual_test/main.cpp	59 match

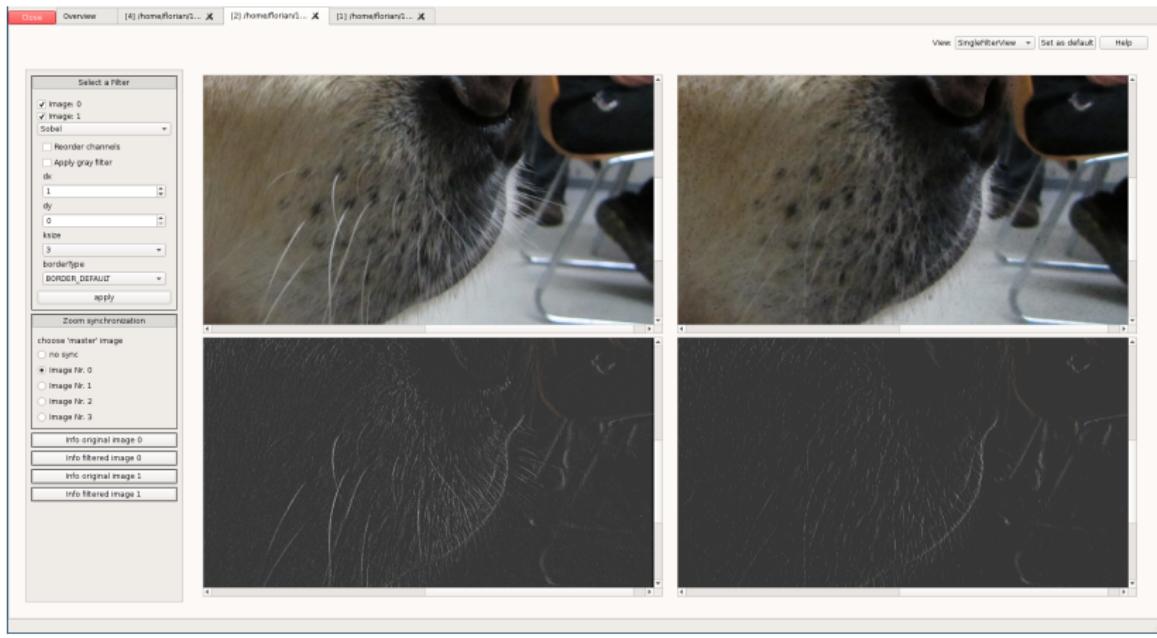
# Filter

- 2 Bilder → 1 Bild
- Differenzbilder, Overlay, geänderte Pixel für Filter



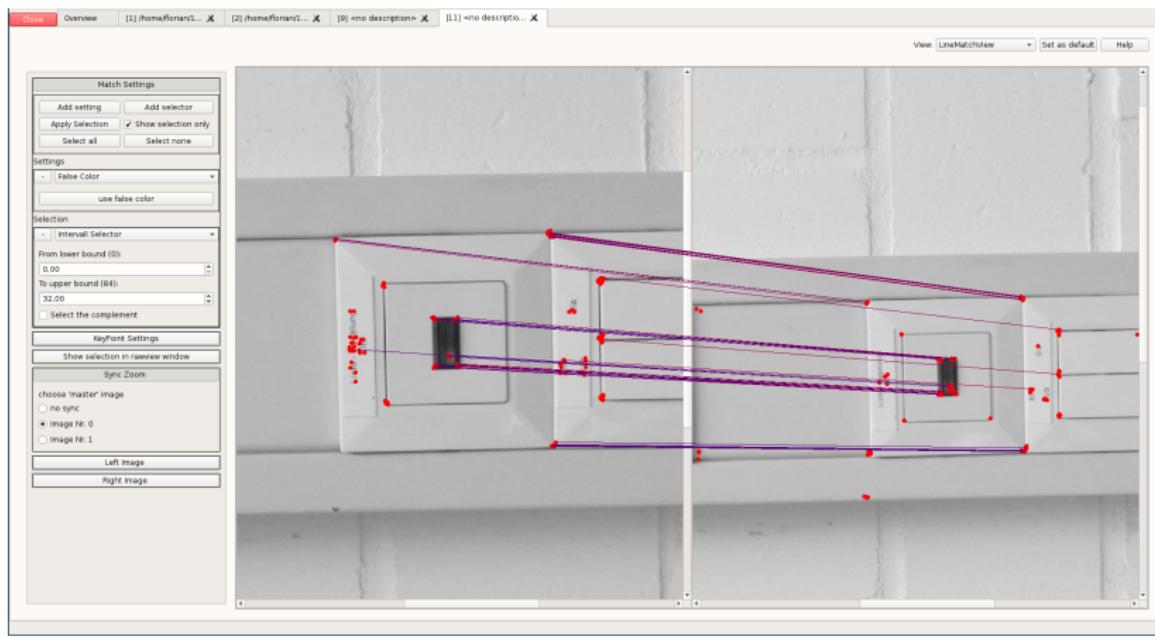
# Filter

- 1 Bild → 1 Bild
- Nachträgliche Anwendung weiterer Filter



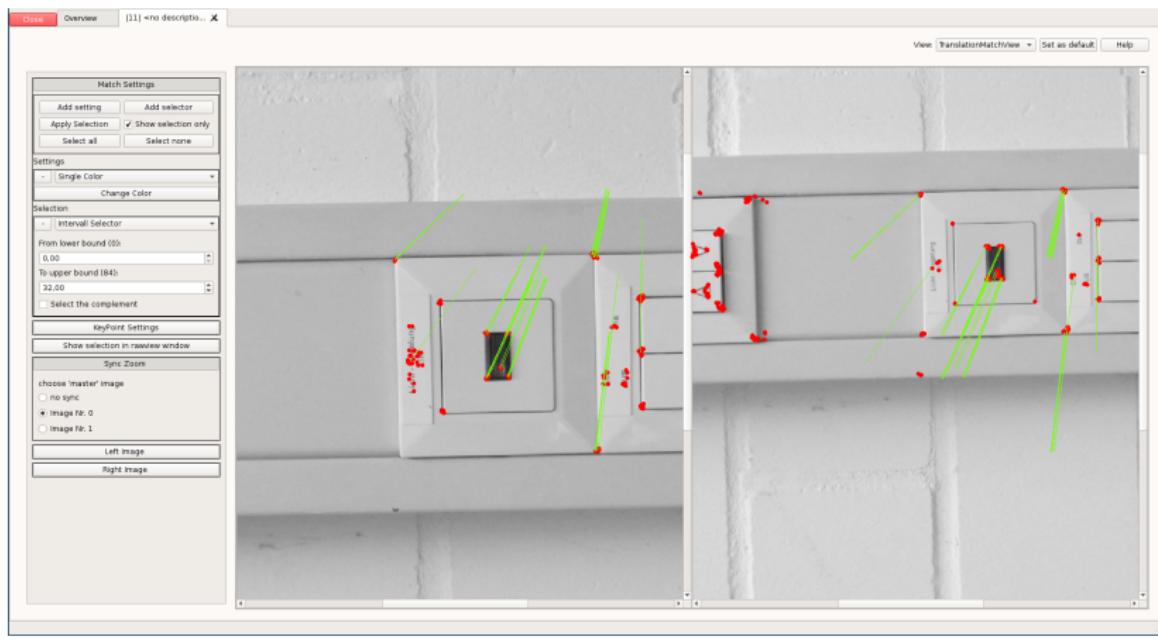
# Matches

- Anzeigen / Filtern von Keypoints / Matches
- Anzeige der Verbindungen von Keypoints



# Matches

- Anzeigen / Filtern von Keypoints / Matches
- Anzeige der Translation von Keypoints

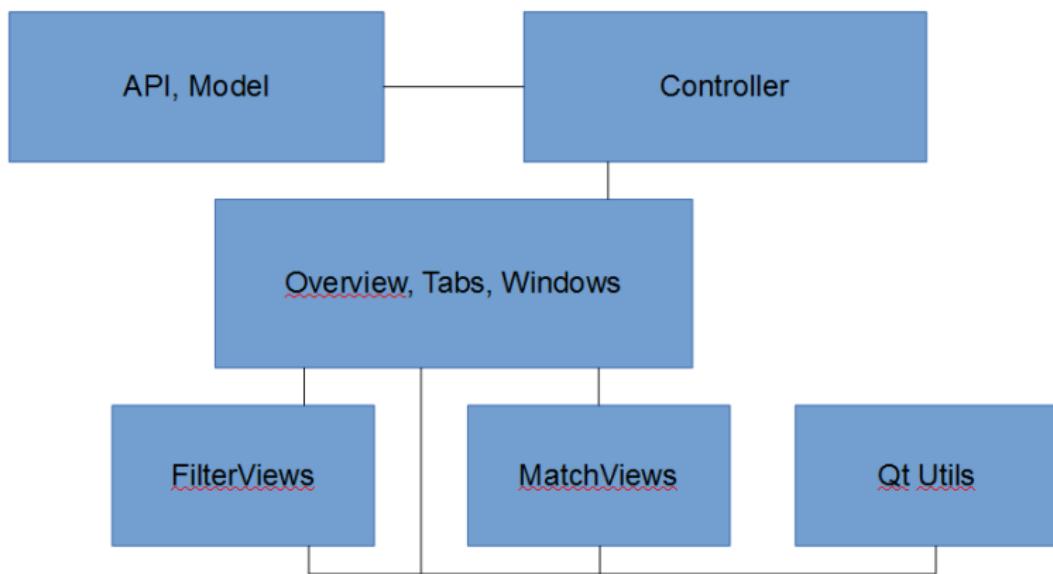


# GUI-Demo

# Architektur

# Entwurf

- Trennung in API, Datenhaltung, Visualisierung



# Signals/Slots & Templates

- Qt erlaubt keine Templateklassen mit Q\_OBJECT
- Signals/Slots in Objekte ausgelagert

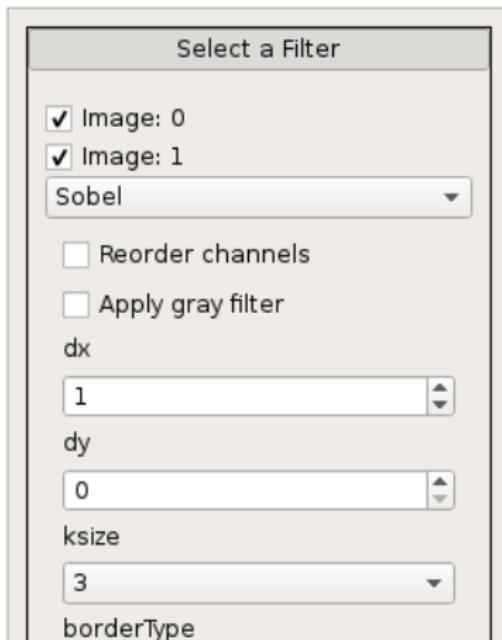
```
class SlotQString::public QObject
{
    Q_OBJECT
public:
    SlotQString(const std::function<void(QString)>& f,
                QObject *parent = nullptr)
        : QObject{parent}, function_{f}
    {
        if (!f)
            throw std::invalid_argument{"invalid function"};
    }
public slots:
    void slot(QString t) const
    {
        function_(t);
    }
private:
    std::function<void(QString)> function_;
};
```

# RegisterHelper

- Ermöglicht die Auswahl von Funktionen über eine Combobox
- Funktionen werden über eine API Funktionen registriert

# (Auto-)FilterWidget

- Unterklasse von RegisterHelper
- Ermöglicht Auswahl von Filtern
- Gibt Ergebnise per Signal weiter (z.B. an ein ZoomableImage)

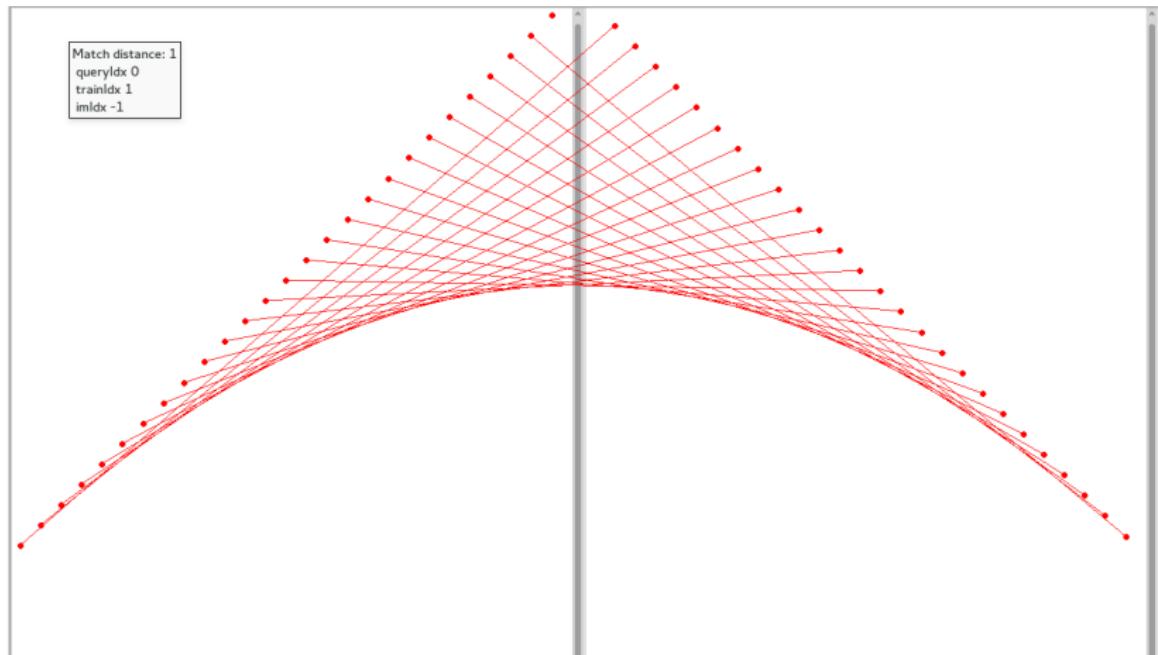


# ZoomableImage

- Umwandlung von cv::Mat in Qt Format
- Signal & Slot für Zoom Events
- Slot zum Bild wechseln
- SyncZoomWidget erlaubt synchrone Zoom
- ZoomableImageOptionPanel zeigt weiter Informationen/Optinen an

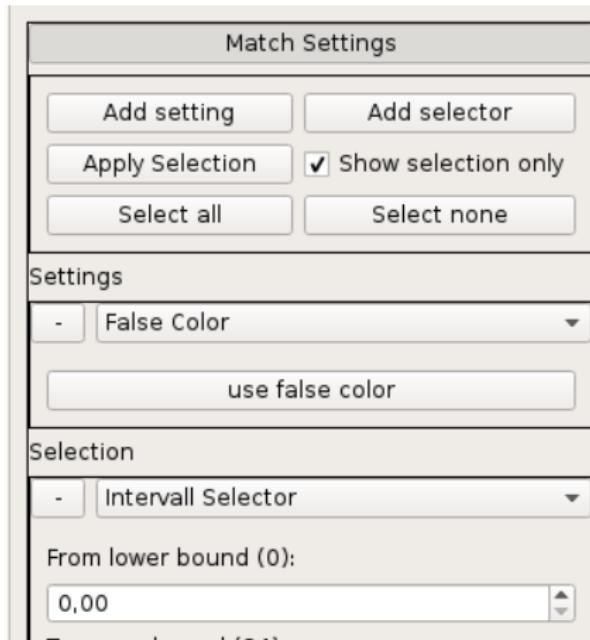
# MatchScene

- Enthält 2 ZoomableImages
- Enthält die KeyPoints/Matches als QGraphicsoObjects



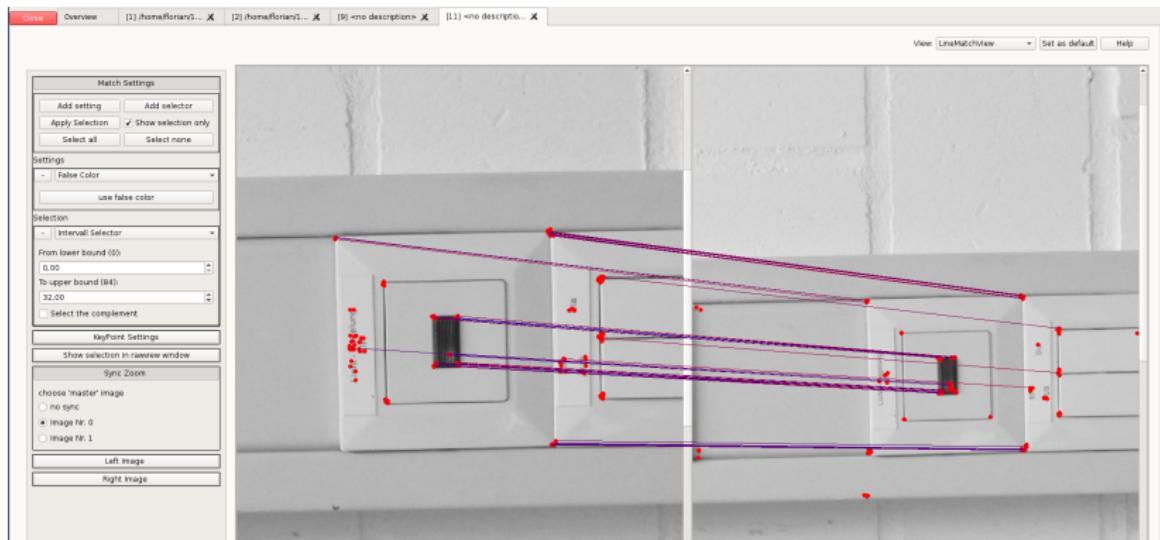
# Match/KeyPointSetting

- Keine Auslagerung von Signals/Slots möglich
- Daher parallele Entwicklung von KeyPoint und MatchSetting
- Nur Selektierte KeyPoints/Matches werden angezeigt



# Views

- Visualisierung der unterschiedlichen Aufrufe
- Unterscheiden sich meist in unterschiedlichen Nutzen von QT Util Klassen
- Einzige Aufgabe Weiterleitung und Annahme der Selektion (beim Wechsel der Views)



# Dokumentation

# Tutorials, Beispiele

The screenshot shows a web browser window titled "CVVisual : CVVisual Example". The address bar contains "cvv.mostlynerdless.de". The main content area displays the "CVVisual Example" page. On the left, there is a sidebar with navigation links: "Home", "TUTORIAL" (with "Introduction to using CVVisual", "Introduction to filter function widgets", and "Über CVVisual"), "REFERENCE" (with "Views" and "Filter query language"), and "API Reference". The main content area has a large heading "CVVisual Example". Below it, a sub-section title "CVVisual Example" is shown. A text block explains that CVVisual is a debug visualization for OpenCV, designed to help programmers see what their code is doing. It includes a code snippet for "code\_example/main.cpp" and notes about includes. Another text block discusses taking snapshots with the webcam. At the bottom, there are two code snippets: one for showing an image and another for converting it to grayscale.

**CVVisual Example**

**CVVisual Example**

CVVisual is a debug visualization for OpenCV, thus, its main purpose is to offer different ways to visualize the results of OpenCV functions to make it possible to see whether they are what the programmer had in mind, and also to offer some functionality to try other operations on the images right in the debug window. This text wants to illustrate the use of CVVisual on a code example.

Image we want to debug this program:

code\_example/main.cpp

Note the includes for CVVisual:

```
10 #include <opencv2/debug_mode.hpp>
11 #include <opencv2/show_image.hpp>
12 #include <opencv2/filter.hpp>
13 #include <opencv2/dmatch.hpp>
14 #include <opencv2/final_show.hpp>
```

It takes 10 snapshots with the webcam. With each, it first shows the image alone in the debug window,

```
97 cvv::showImage(imgRead, CVVISUAL_LOCATION, imgIdString.c_str());
```

then converts it to grayscale and calls CVVisual with the original and resulting image,

```
101 cv::cvtColor(imgRead, imgGray, CV_BGR2GRAY);
```

# Kurzdokumentation

Wird von der Hilfefunktion des Programms benutzt.

**CVVisual** OpenCV debug visualization

[Home](#)

TUTORIAL

[Introduction to using  
CVVisual](#)

[Introduction to filter  
function widgets](#)

[Über CVVisual](#)

REFERENCE

[Views](#)

[Filter query language](#)

[API Reference](#)

## Views

### General information:

Most views offer an `ImageInformation` collapsable in their accordion menus.

The zoom can be found here.

`Ctrl + Mouse wheel` is also ZOOM; `Ctrl + Shift + Mouse wheel` is a slower zoom.

If the zoom is deeper than 60%, the image's pixels will be overlaid with their channel values; usually, the order is `BGR[+alpha]` from the top.

### Single Image View:

Associated with the `debugSingleImage()` function.

Shows one single image with no features other than `Image Information`.

### Filter Views:

Associated with the `debugFilter()` function.

### DefaultFilterView:

Shows two images with only the basic features of `ImageInformation`, synchronized zoom and `Histogram`.

### DualFilterView:

Shows the two images given to the `CVVisual` function and `Result Image` inbetween which represents the result of a filter that was applied to the others via the `Filter selection` collapsable, like a difference image between the two.

# Referenz:

- Mit Hilfe von Doxygen

The screenshot shows a web browser window with the title "CVVisual: cv::qutill::Accordion Cl..." and the URL "cvv.mostlynerdless.de/api/classcvv\_1\_1qutill\_1\_1Accordion.html". The page content is a Doxygen-generated API documentation for the `Accordion` class.

**CVVisual**  
A debug visualization for opencv

Main Page Namespaces Classes Files Search

Class List Class Index Class Hierarchy Class Members

**Public Member Functions**

- `Accordion (QWidget *parent=nullptr)`  
Constructs an empty accordion. More...
- `void clear ()`  
Removes all elements and deletes them immediately. More...
- `void collapse (Handle handle, bool b=true)`  
Collapses an element. More...
- `void collapseAll (bool b=true)`  
Collapses all elements. More...
- `Collapsible & element (Handle handle)`  
Returns the element corresponding to handle. More...
- `const Collapsible & element (Handle handle) const`
- `void expand (Handle handle, bool b=true)`  
Expands an element. More...
- `void expandAll (bool b=true)`  
Expands all elements. More...
- `void hide (Handle handle, bool b=true)`

cvv qutill Accordion

Generated on Tue Mar 25 2014 22:45:17 for CVVisual by doxygen 1.8.6

# API

# Anwender API

- Triviale Benutzung auch in C++98
- Sehr klein und übersichtlich

# Interne API

- Erweiterung über Funktionen in `cvv::extend`
- Leichtes, zentralisiertes Hinzufügen von Visualisierungen, Filtern, Views,...

# Ausblick

# Rezeption

Projekt schien von der OpenCV-Community wohlwollend aufgenommen zu werden



snosov1 commented 2 days ago

Collaborator

Hi, Andreas!

First of all, thank you for a really valuable contribution. I've been dreaming about such functionality since the day 1 I started using OpenCV.

As @apavlenko suggests, this module should probably go to the opencv\_contrib repository. Due to limited resources we've created it, so we could easily accept such big PRs - almost "No questions asked". Then it boils there for a bit of time, and if it turns out to be solid and well received by the community, we would merge it into the mainstream (this) repo.

It's a default path for such major contributions and if you're ok with it - let's do it this way.

Personally, I would like such module to be in the mainstream repo as soon as possible. So, I'll try to review it shortly and give some feedback.

# Rezeption

Nach aktuellem Stand aber aufgrund C++11 und Qt5 keine Aufnahme ins Haupt-Repo



snosov1 commented on 19. Apr.

Sorry for delay. I've looked through it right away, and there're a couple of issues. Mainly, we don't plan to enable C++11 for builds of this repository, since the support is not yet ubiquitous. Also, the usage of Qt5 is rather limiting.

This makes it a great tool for development and research on Desktops with latest sw, but is unusable on other platforms.

My thinking is that in its current form it doesn't belong to the mainstream repo because of these dependencies. But, I think, it can be merged to the contrib repo after a few minor fixes.

Let's also ask [@kirill-konyakov](#) on that.

# Links

- Github: <https://github.com/CVVisualPSETeam/CVVisual/>
- Dokumentation: <https://cvv.mostlynerdless.de/>
- Doxygen: <https://cvv.mostlynerdless.de/api/>