Digital Image Processing

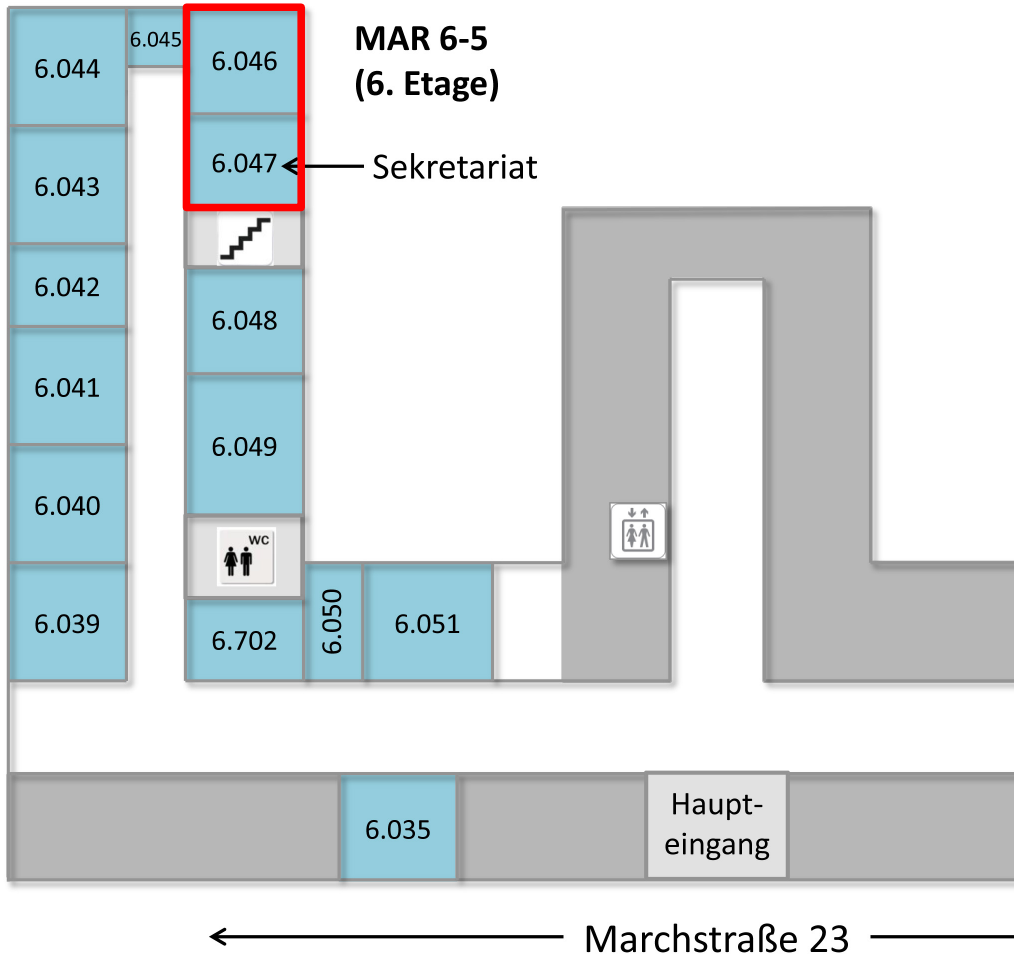
Berlin University of Technology (TUB),
Computer Vision and Remote Sensing Group
Berlin, Germany

Ronny Hänsch

- **E-Mail:**
  r.haensch@tu-berlin.de

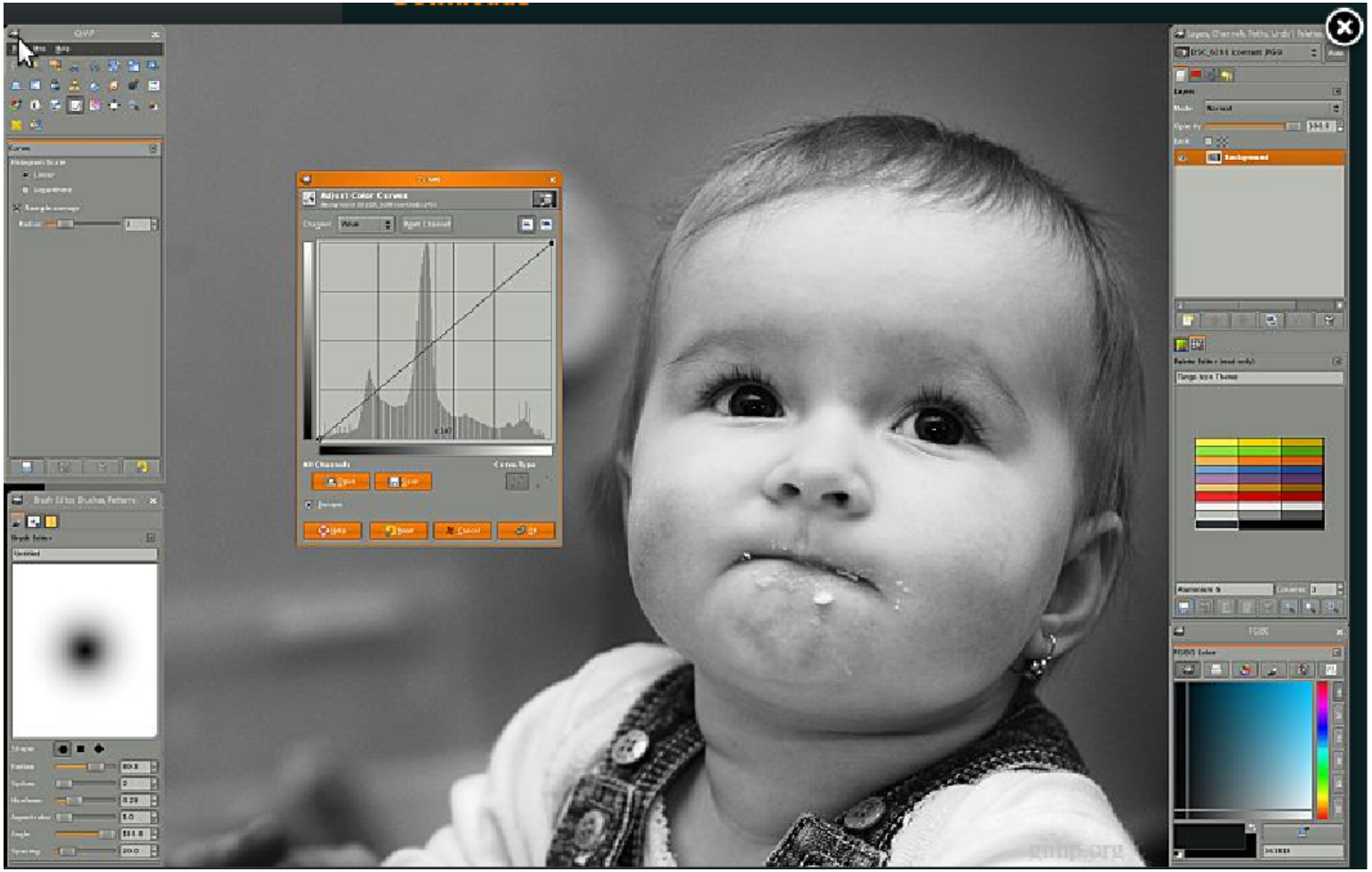
Olaf Hellwich

- **E-Mail:** olaf.hellwich@tu-berlin.de
- **Office**
  - MAR6.046, March Building, 6th Floor
- **Consultation Time**
  - Thursday, 13:00-15:00 o'clock
  - Only by arrangement marion.dennert@tu-berlin.de



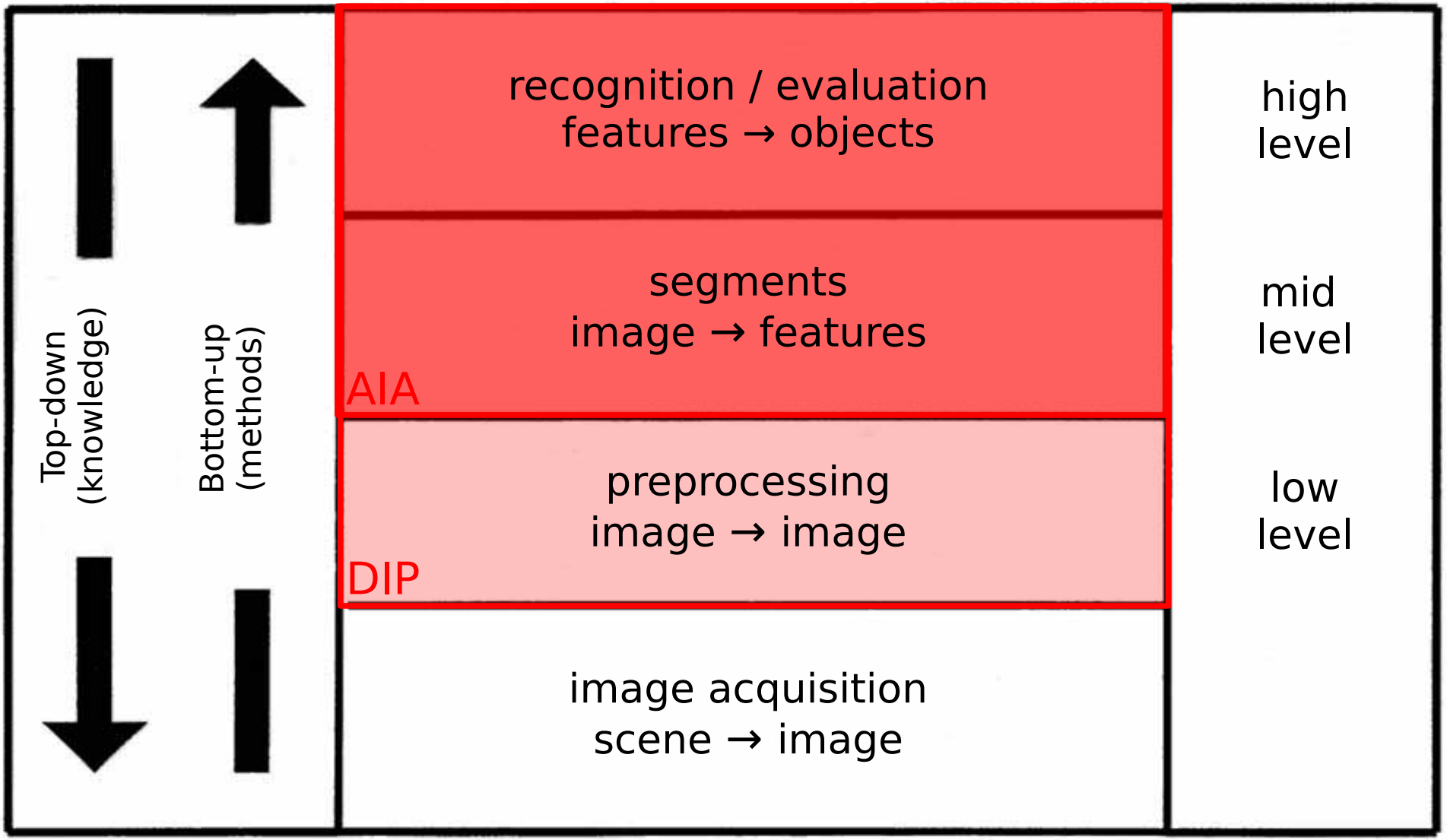
What are you gonna learn?

Photoshop, Gimp, …



- NOT how to USE it (image editing)
- BUT how it WORKS (image processing)

What are you gonna learn?



How are you gonna learn it?

1. Visit lectures
   - Every week (HL 001, Tuesday, 10-12 o'clock)

2. Visit exercises
   - Every two weeks (HL 001, Friday, 10-12 o'clock)

3. Doing homework
   - Consultation time: MAR6.046, Thursday, 13:00-15:00 o'clock
     Only by arrangement: marion.dennert@tu-berlin.de

4. ASK QUESTIONS!
   - As often as possible

5. (Read further material)
   - As often as possible

Material

Books

- Petrou: **Image Processing – The Fundamentals**
- Gonzalez, Woods: **Digital Image Processing**
- Jähne: **Digital Image Processing**
- Sonka et al.: **Image Processing, Analysis, and Machine Vision**

Articles

- Scientific paper: www.ieeexplore.com
  (free download within TU-network)

Common mistakes made by students

IEEE Potentials,
M.N.O. Sadiku, S.M. Musa, K. Kirby,
July/August 2012

"It is wise and better to learn from other people's mistakes and avoid them."

→ 7 common mistakes
made by engineering students
and how to avoid them

IMAGE COURTESY OF STOCK.XCHNG/ZORAN OZETSKY

Common mistakes made by students

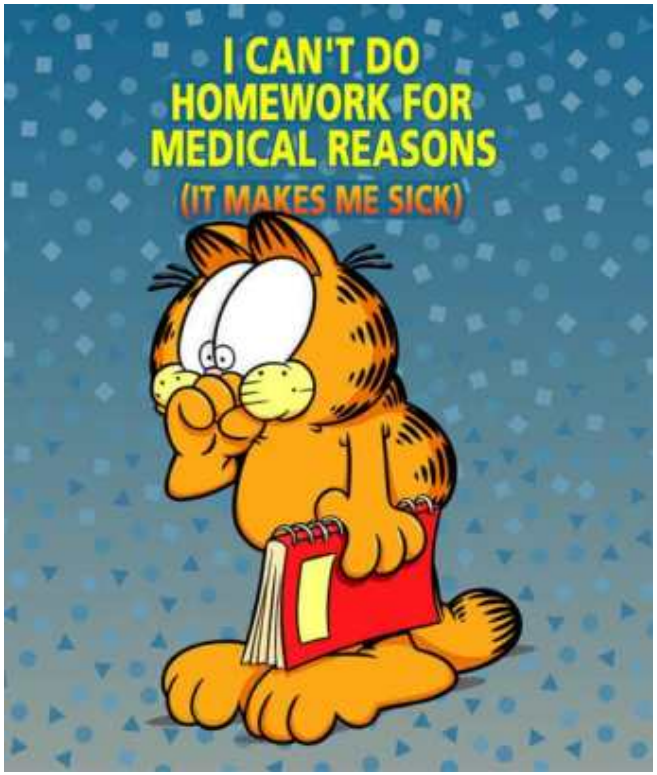
1. Not attending classes

- Possible reasons:
  - Unavoidable circumstances (e.g. collision with other courses)
  - Lazy...
  - Professor does not teach well
    → “I can just read the textbook”

- Recommendation:
  - Attend class and be punctual
    → You can always gain something

Common mistakes made by students

2. Not doing the homework



- Possible reasons:
  - Too boring
  - Too hard
  - No understanding what to do
  - No time (matter of priorities!)

- Recommendation:
  - Do the homework yourself (together with group members)
  - No cheating (e.g. copying from others)
    → Problem solving is essential for learning process

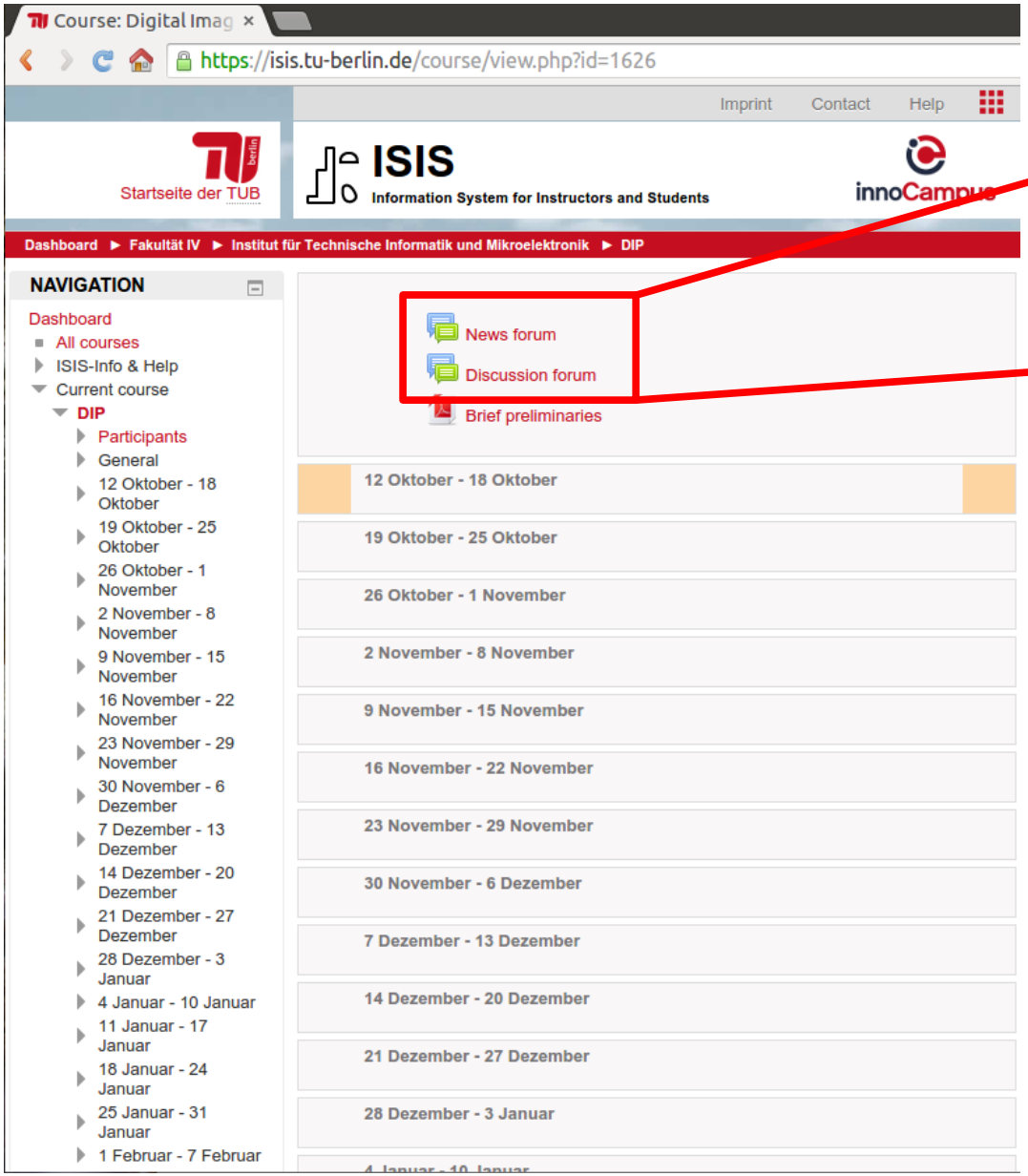
Informations

WWW

- Information, important announcements:
  - → http://www.cv.tu-berlin.de Announcements: 'Lectures'

  - → Slides and other material: **ISIS2**

Information



- Open for discussions about
  - → Image processing topics
  - → General computer vision
  - → Homework, eg. Installation advices, bugs in provided code, etc.

- Not the place to ask questions to Prof. Hellwich! → EMail

Exams

- **Mid-term:**
  - Near the middle of the term: tba
  - Room: tba
  - Duration: 45 min
  - In place of an exercise
  - No grade, but pass is necessary to take part at the final exam

- **Final:**
  - At the end of the term: **21.02.2018**
  - Room: HE101
  - Duration: 12-14 o'clock (90 min)

- Questions in English, answers in English or German

Homework

What to do?

I: Answering theoretic questions

II: Implementation of methods for processing digital images

How?

In groups of 3-4 students

Programming Language: C++ [and OpenCV **2.4** or **3.0**]

Completion of provided software packages

- Class descriptions (header files): given
- Includes: given
- Basic functionality: given
- Specific functions: Your task!

Goal?

Practising, Learning. No grades!

But pass is necessary to take part at the final exam

Homework

Group Work

- Groups consist of 3-4 students

- Group selection via ISIS (already open)

- Groups with less than 3 students will be merged

- Deadline: Next Thursday, 26.10.

- Midterm is individual work

Homework

Group Work

Homework

- Next meeting in two weeks

- **BEFORE Friday, 10am**:
  - Hand in your solution via ISIS

  - Submission includes (red denotes **mandatory** material):
    → **All program files** of the provided material
      - i.e. all .h and .cpp files, no executables!
    → **Input**, intermediate, and **output images**
    → **Pdf-file with**
      → Short discussion / presentation of your solution
      → **Answers to theoretical questions**
  - Submission DOES NOT include
    → Executable, project files, etc.

- Algorithms more important than well-written code (but try!)

Homework

- “Grades”

+++	more than just a correct solution (efficient, clever, cool, …)
++	correct solution
+	some minor errors, but still acceptable
-	not acceptable → re-work *(within 1 week, parallel to new assignment!)*
- -	failed: you are not allowed to write the exam!

1. Exercise – Part I : Theory

1. What is a digital image?

2. What does the paradigm “bottom-up processing” mean?

3. State at least three fundamentally different image sources!

1. Exercise – Part II : Practical

C++ and OpenCV

Given:

- Main function (main.cpp)
- Function declaration (Dip1.h)
- Basic functionality (Dip1.cpp)

Todo:

- **[Install C++-compiler]**
- **[Install OpenCV]**
- Dip1.cpp
  - Mat Dip1::doSomethingThatMyTutorIsGonnaLike(Mat&)
    → Do something (reasonable)

Deadline:

- Next meeting at **03.11.2017**, 10am

1. Exercise – Part II : Given

FILE: main.cpp

```
int main(int argc, char** argv)
```

- Main function
  - Usage:
    - dip1 path_to_image
      - Calls Dip1::run(...)
      - Calls Dip1::test(...) for basic testing!

FILE: Dip1.cpp/h

```
void Dip1::run(string fname)
```

- Loads image
- Calls related functions

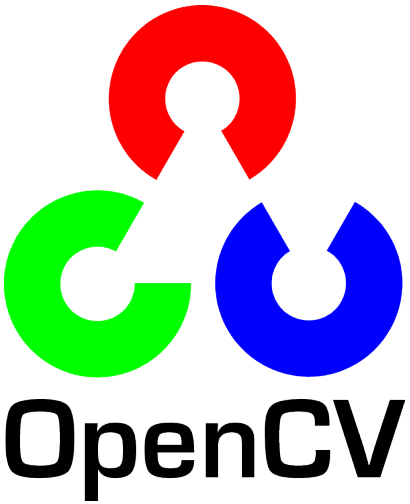
1. Exercise – Part II : To Do

```
Mat doSomethingThatMyTutorIsGonnaLike(Mat& img)

img      :   input image
return   :   output image
```

→ does something cool... (hopefully)

Brief introduction to OpenCV



- One of most common image processing libs
- Open source (BSD)
- C/C++, Python, Java interfaces
- Supported: Windows, Linux, Mac OS, iOS, Android
- Strong focus on real-time applications
- Multi-core processing, hardware acceleration
- 47 thousand people in user community
- 9 million downloads

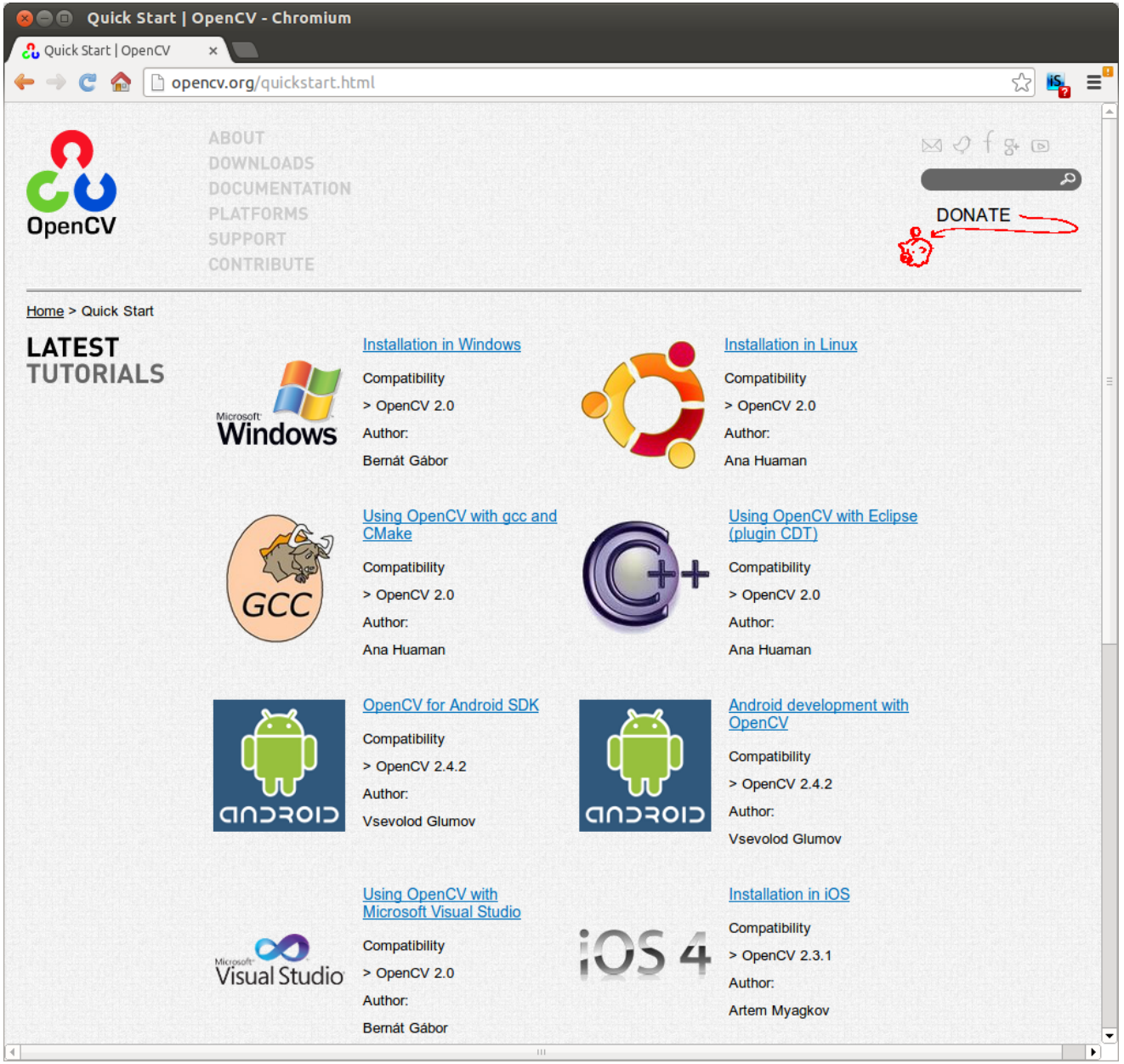
Brief introduction to OpenCV

OpenCV API Reference

- Introduction
  - API Concepts
- core. The Core Functionality
  - Basic Structures
  - Basic C Structures and Operations
  - Dynamic Structures
  - Operations on Arrays
  - Drawing Functions
  - XML/YAML Persistence
  - XML/YAML Persistence (C API)
  - Clustering
  - Utility and System Functions and Macros
  - OpenGL interoperability
- imgproc. Image Processing
  - Image Filtering
  - Geometric Image Transformations
  - Miscellaneous Image Transformations
  - Histograms
  - Structural Analysis and Shape Descriptors
  - Motion Analysis and Object Tracking
  - Feature Detection
  - Object Detection
- highgui. High-level GUI and Media I/O
  - User Interface
  - Reading and Writing Images and Video
  - Qt New Functions
- video. Video Analysis
  - Motion Analysis and Object Tracking
- calib3d. Camera Calibration and 3D Reconstruction
  - Camera Calibration and 3D Reconstruction
- features2d. 2D Features Framework
  - Feature Detection and Description
  - Common Interfaces of Feature Detectors
  - Common Interfaces of Descriptor Extractors
  - Common Interfaces of Descriptor Matchers
  - Common Interfaces of Generic Descriptor Matchers
  - Drawing Function of Keypoints and Matches
  - Object Categorization
- objdetect. Object Detection
  - Cascade Classification
  - Latent SVM
- ml. Machine Learning
  - Statistical Models
  - Normal Bayes Classifier
  - K-Nearest Neighbors
  - Support Vector Machines
  - Decision Trees
  - Boosting
  - Gradient Boosted Trees
  - Random Trees
  - Extremely randomized trees
  - Expectation Maximization
  - Neural Networks
  - MLData
- flann. Clustering and Search in Multi-Dimensional Spaces
  - Fast Approximate Nearest Neighbor Search
  - Clustering
- gpu. GPU-accelerated Computer Vision
  - GPU Module Introduction
  - Initalization and Information
  - Data Structures
  - Operations on Matrices
  - Per-element Operations
  - Image Processing
  - Matrix Reductions
  - Object Detection
  - Feature Detection and Description
  - Image Filtering
  - Camera Calibration and 3D Reconstruction
  - Video Analysis
- photo. Computational Photography
  - Inpainting
  - Denoising
- stitching. Images stitching
  - Stitching Pipeline
  - References
  - High Level Functionality
  - Camera
  - Features Finding and Images Matching
  - Rotation Estimation
  - Autocalibration
  - Images Warping
  - Seam Estimation
  - Exposure Compensation
  - Image Blenders
- nonfree. Non-free functionality
  - Feature Detection and Description
- contrib. Contributed/Experimental Stuff
  - Stereo Correspondence
  - FaceRecognizer Documentation
  - Retina Documentation
  - OpenFABMAP
- legacy. Deprecated stuff
  - Motion Analysis
  - Expectation Maximization
  - Histograms
  - Planar Subdivisions (C API)
  - Feature Detection and Description
  - Common Interfaces of Descriptor Extractors
  - Common Interfaces of Generic Descriptor Matchers
- ocl. OpenCL-accelerated Computer Vision
  - OpenCL Module Introduction
  - Data Structures and Utility Functions
  - Data Structures
  - Operations on Matrics
  - Matrix Reductions
  - Image Filtering
  - Image Processing
  - ml.Machine Learning
  - Object Detection
  - Feature Detection And Description
  - Video Analysis
  - Camera Calibration and 3D Reconstruction
- superres. Super Resolution
  - Super Resolution
- viz. 3D Visualizer
  - Viz
  - Widget

Brief introduction to OpenCV

How to install....



Brief introduction to OpenCV

How to install.... on Windows:

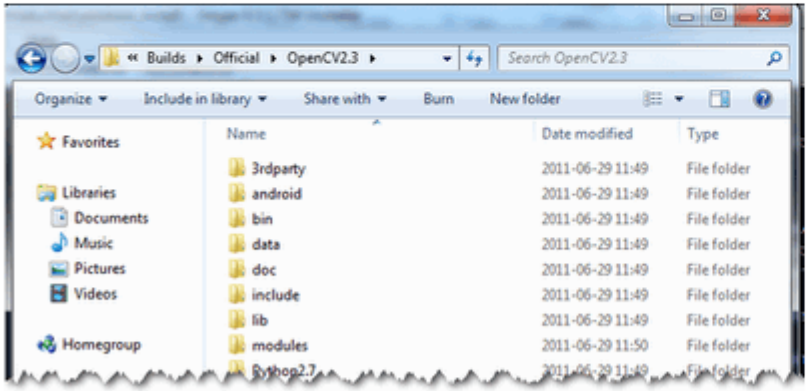
→ Use pre-built libraries (unless you exactly know what you do)

→ Don't forget to set the OpenCV environment
and add it to the system path (if necessary)

→ http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html

Installation by Using the Pre-built Libraries

1. Launch a web browser of choice and go to our page on Sourceforge.
2. Choose a build you want to use and download it.
3. Make sure you have admin rights. Unpack the self-extracting archive.
4. You can check the installation at the chosen path as you can see below.



5. To finalize the installation go to the *Set the OpenCV enviroment variable and add it to the systems path* section.

Brief introduction to OpenCV

How to install.... on Windows with MS Visual Studio:

→ MSVisualStudio freely available at Microsoft Imagine (Fak-IV only)

https://irb.eecs.tu-berlin.de/imagine/

→ http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_visual_studio_Opencv/windows_visual_studio_Opencv.html#windows-visual-studio-how-to

→ See also the section on how to set command line arguments!

→ Alternatives
- → Eclipse CDT https://eclipse.org/cdt/
- → MS Visual Studio Community: www.visualstudio.com

Brief introduction to OpenCV

How to install.... on Mac:

→ No pre-built libraries, build from source
Xcode, CMake

→ http://docs.opencv.org/2.4/doc/tutorials/introduction/ios_install/ios_install.html

→ Tutorials:
http://docs.opencv.org/2.4/doc/tutorials/ios/table_of_content_ios/table_of_content_ios.html

Brief introduction to OpenCV

How to install.... on Linux:

→ Often in the repository
    → maybe not newest version, but it will be sufficient!

→ If you know what to do: Build from source

→ http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html

Brief introduction to OpenCV

How to install.... further information:

→ General: http://opencv.org/
→ Tutorials: http://docs.opencv.org/2.4/doc/tutorials/tutorials.html
→ Docu: http://docs.opencv.org/2.4.13/

→ www.giyf.com

→ Discussion forum on ISIS

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

Includes all opencv headers

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

To use opencv namespace.
Otherwise put cv:: in front
of all opencv functions etc.

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

Reads an image from the path provided in the first command line argument

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( "example");
    imshow( "example", img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite("coolResult.png", newImg);

    waitKey(0);
}
```

Creates a window with the ID "*example*"

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

Displays the content of *img* in the window “*example*”

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

Creates a matrix of same size as *img* containing one channel of 32bit floats

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

The use of smart pointers in Mat allows to use function arguments as output

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

Writes image to disk

Brief introduction to OpenCV

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main(int argc, char** argv){

    Mat img = imread( argv[1] );

    // show image
    namedWindow( “example”);
    imshow( “example”, img);

    Mat newImg( img.rows, img.cols, CV_32FC1 );
    // do something fancy
    fancyFunction(img, newImg);

    imwrite(“coolResult.png”, newImg);

    waitKey(0);
}
```

Stops execution until key is pressed

Brief introduction to OpenCV

Matrix generation, some examples:

```
Mat M1 = Mat(2, 3, CV_32FC1);        // creates 2x3 matrix of floats (one channel)
Mat M2 = Mat(3, 2, CV_64FC2);        // creates 3x2 matrix of doubles (two channels)
Mat M3 = Mat(3, 3, CV_8UC3);         // creates 3x3 matrix of uint (three channels)

Mat M4 = Mat::zeros(3, 3, CV_32FC1);     // creates 3x3 matrix of floats, all set to 0
Mat M5 = Mat::ones(3, 3, CV_32FC1);      // creates 3x3 matrix of floats, all set to 1

Mat M6 = (Mat_<float>(3,3) << 1, 2, 3, 4, 5, 6, 7, 8, 9);
```

Accessing matrix data (the easy way)

```
M1.at<float>(row, column)      =    22.0 / 7.0;
M2.at<Vec2d>(row, column)   =    Vec2d(0,1);
int s = M3.at<Vec3b>(row, column)[0];
```

Brief introduction to OpenCV

Accessing Image data – The hard way

```
float sum( Mat& img ){

    float s = 0.0;

    for(int y=0; y < img.rows; y++){

        uchar* data = img.ptr<uchar>(y);

        for(int x=0; x < img.cols; x++ ) {
            s += data[x];
        }
    }
    return s;
}
```

Brief introduction to OpenCV

Compilation (Linux etc.)

```
user@comp:~/path$ g++ -o dip dip1.cpp main.cpp -lopencv_core -lopencv_imgproc -lopencv_highgui
user@comp:~/path$ g++ -o dip dip1.cpp main.cpp `pkg-config opencv --cflags –libs`
```

Or using *cmake* and *make*

Further information:

- http://opencv.org/
  - Install guides
  - Documentation
  - FAQ

1. Exercise

C++ and OpenCV

OpenCV-functions, that might be useful:

- Images I/O:
  - `imread(...), imwrite(...)`
- Color conversion:
  - `cvtColor(...), e.g. CV_BGR2GRAY`
- Type conversion:
  - `M.convertTo(...)`
- Matrix creation:
  - `Mat(...), Mat::zeros(...), Mat::eye(...)`
- Setting/Getting elements of a matrix:
  - `M.at<T>(...)`
- Matrix multiplication:
  - `M1 * M2`
- Matrix multiplication (component-wise):
  - `M1.multiply(M2)`