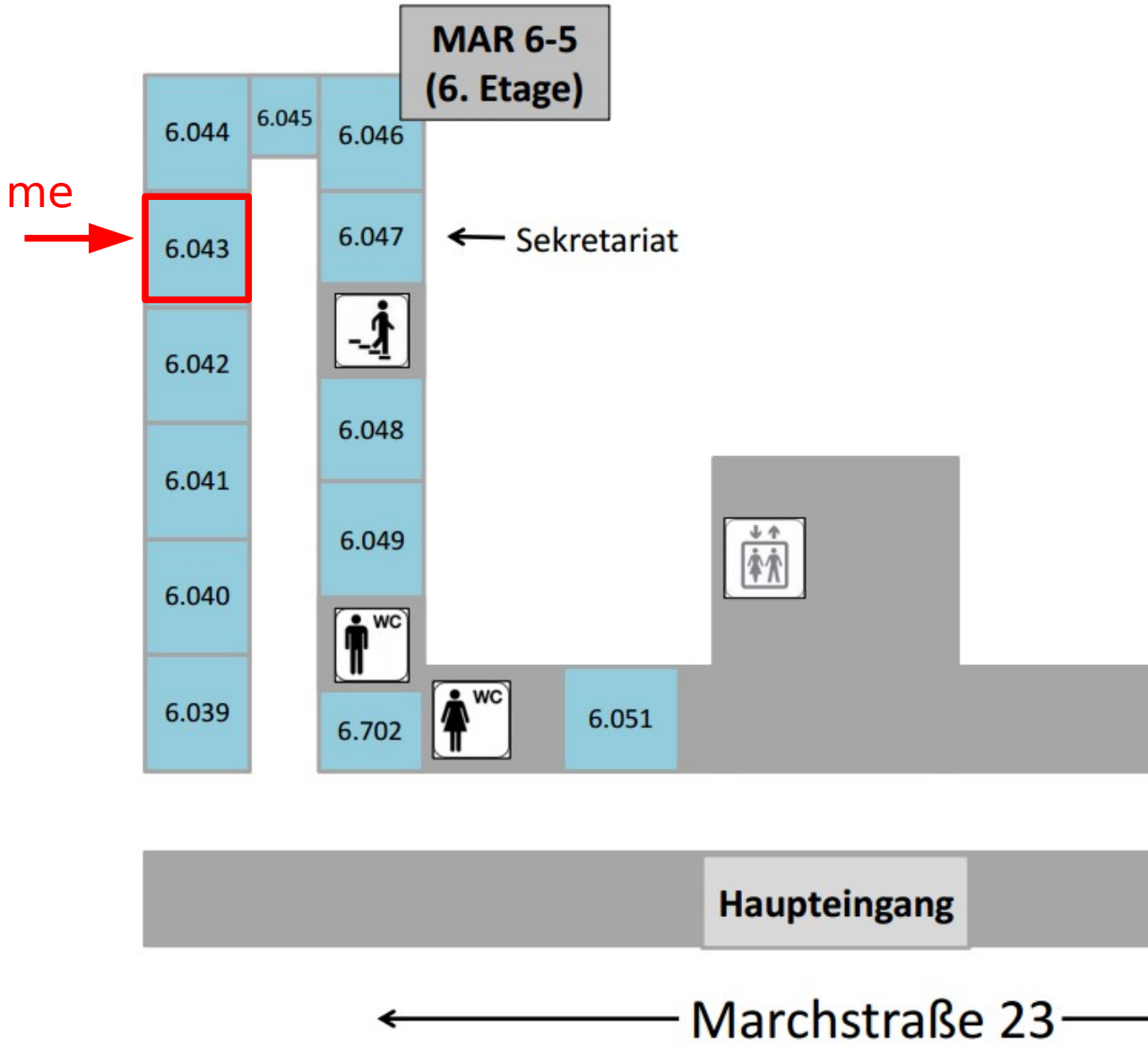
Digital Image Processing

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

Contact

Andreas Ley

- **E-Mail:**
  - ➔ andreas.ley@tu-berlin.de
- **Office**
  - ➔ MAR6.043,
    March Building, 6th Floor
- **Consultation Time**
  - ➔ Wednesday,
    16:30-17:30 o'clock



Teaching

- **Digital Image Processing**
  - Image → Image
  - Image → Description
  - Winter Term

- **Automatic Image Analysis**
  - Image → Object Model
  - Image → Object Detection
  - Summer Term

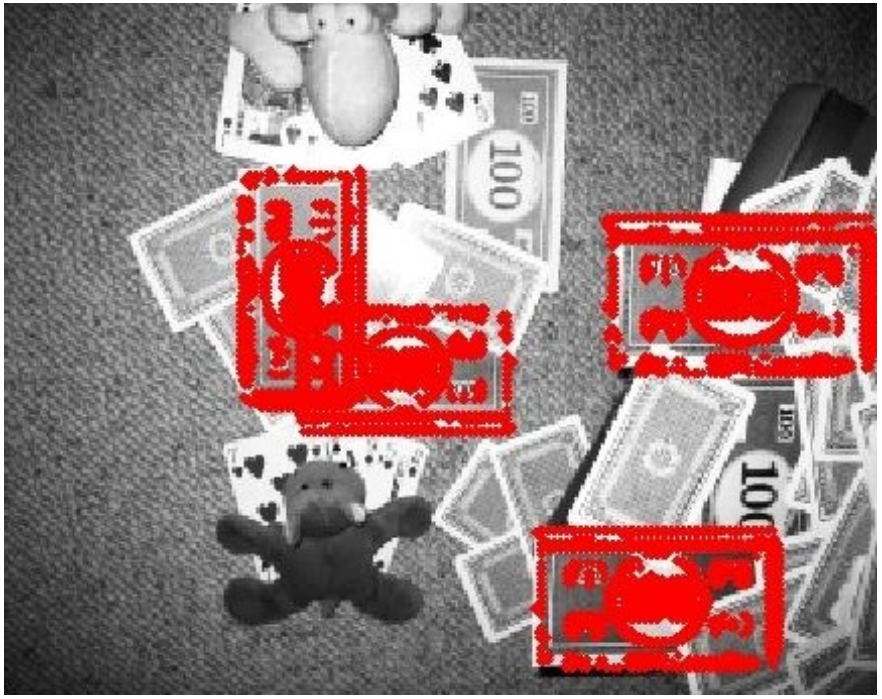
- **Photogrammetric Computer Vision**
  - Image(s) → 3D Model
  - Winter Term

- **Projects/Seminars**
  - Summer Term
  - e.g.: Scientific Process in Computer Vision

Teaching

- **Digital Image Processing**
  - Image → Image
  - Image → Description
  - Winter Term

- **Automatic Image Analysis**
  - Image → Object Model
  - Image → Object Detection
  - Summer Term



- **Photogrammetric Computer Vision**
  - Image(s) → 3D Model
  - Winter Term

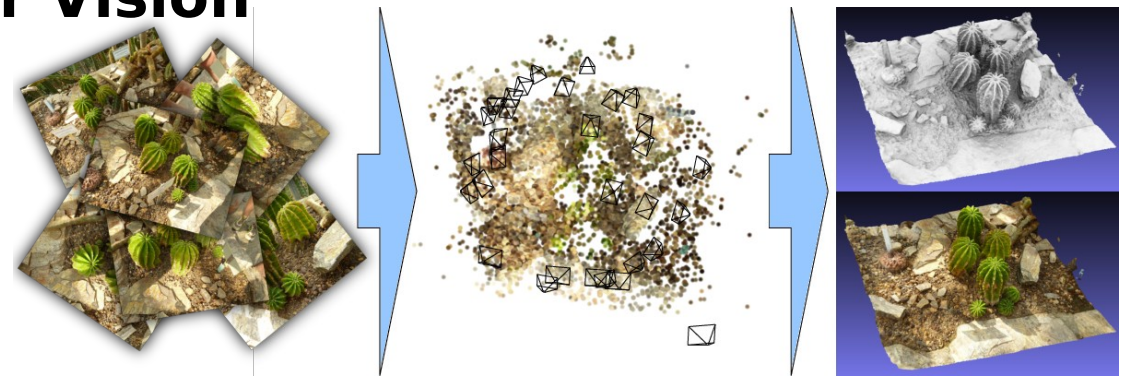
- **Projects/Seminars**
  - Summer Term
  - e.g.: Scientific Process in Computer Vision

Teaching

- **Digital Image Processing**
  - Image → Image
  - Image → Description
  - Winter Term

- **Automatic Image Analysis**
  - Image → Object Model
  - Image → Object Detection
  - Summer Term

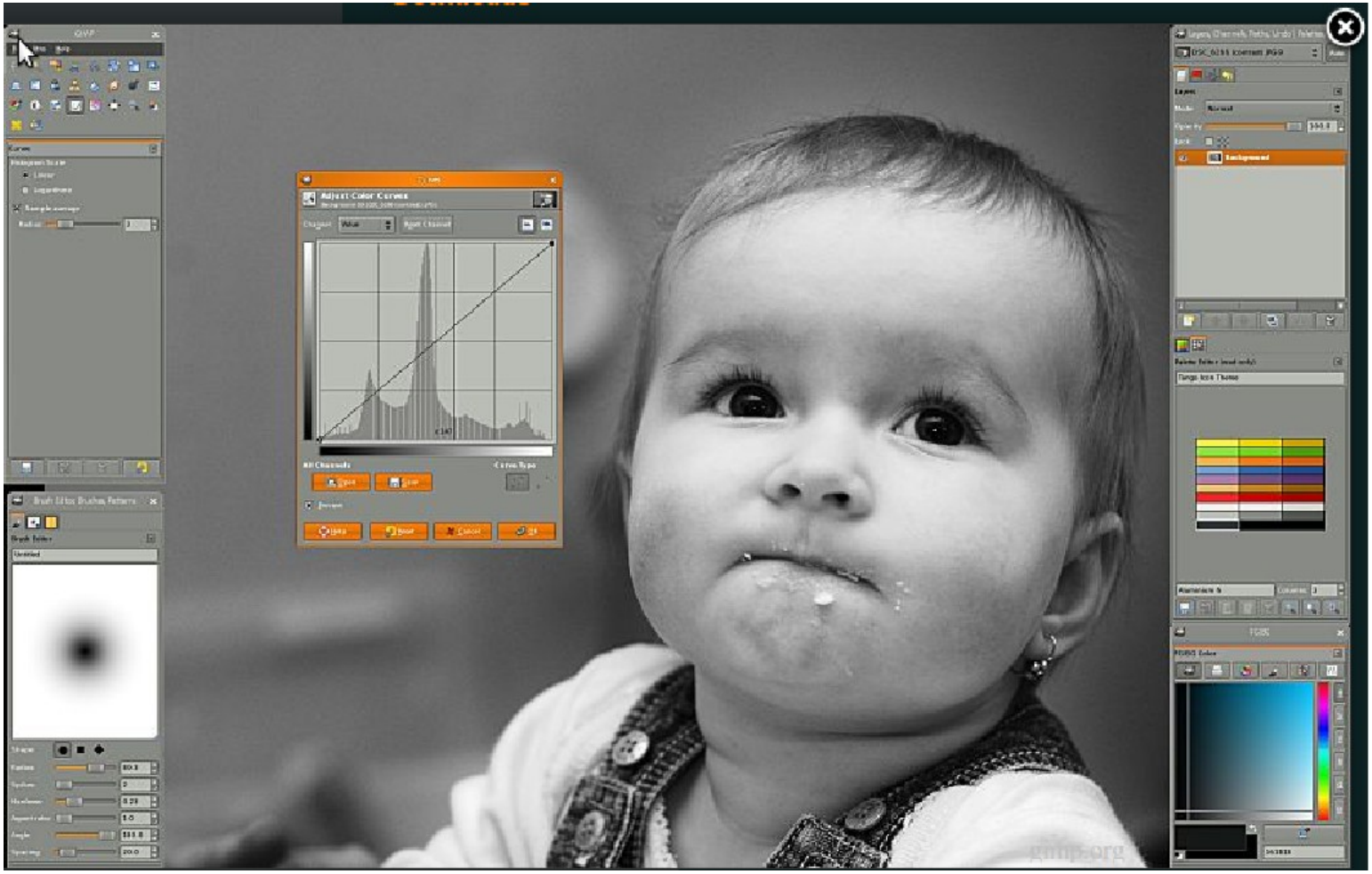
- **Photogrammetric Computer Vision**
  - Image(s) → 3D Model
  - Winter Term



- **Projects/Seminars**
  - Summer Term
  - e.g.: Scientific Process in Computer Vision

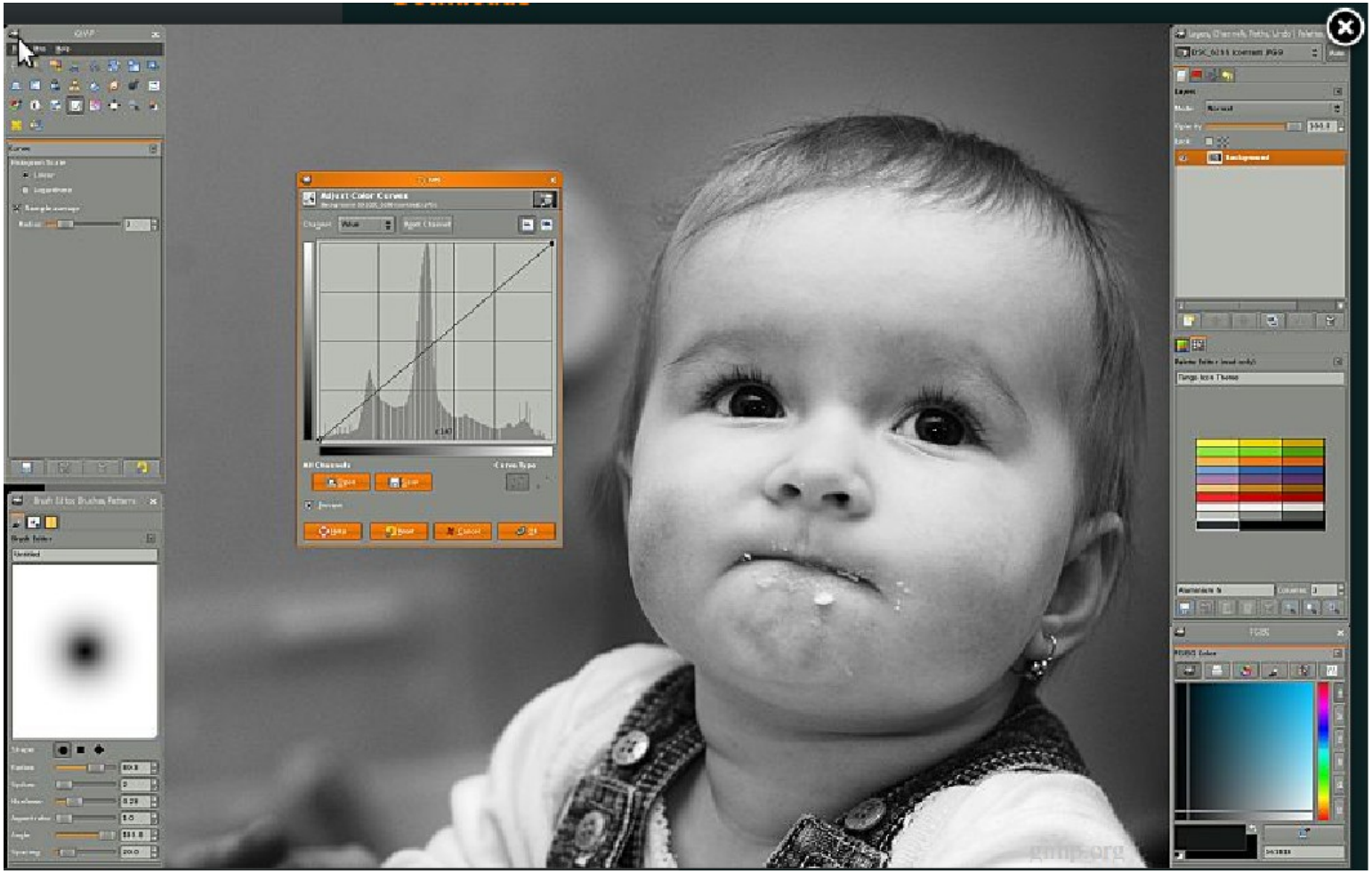
What are you gonna learn?

Photoshop, Gimp, …



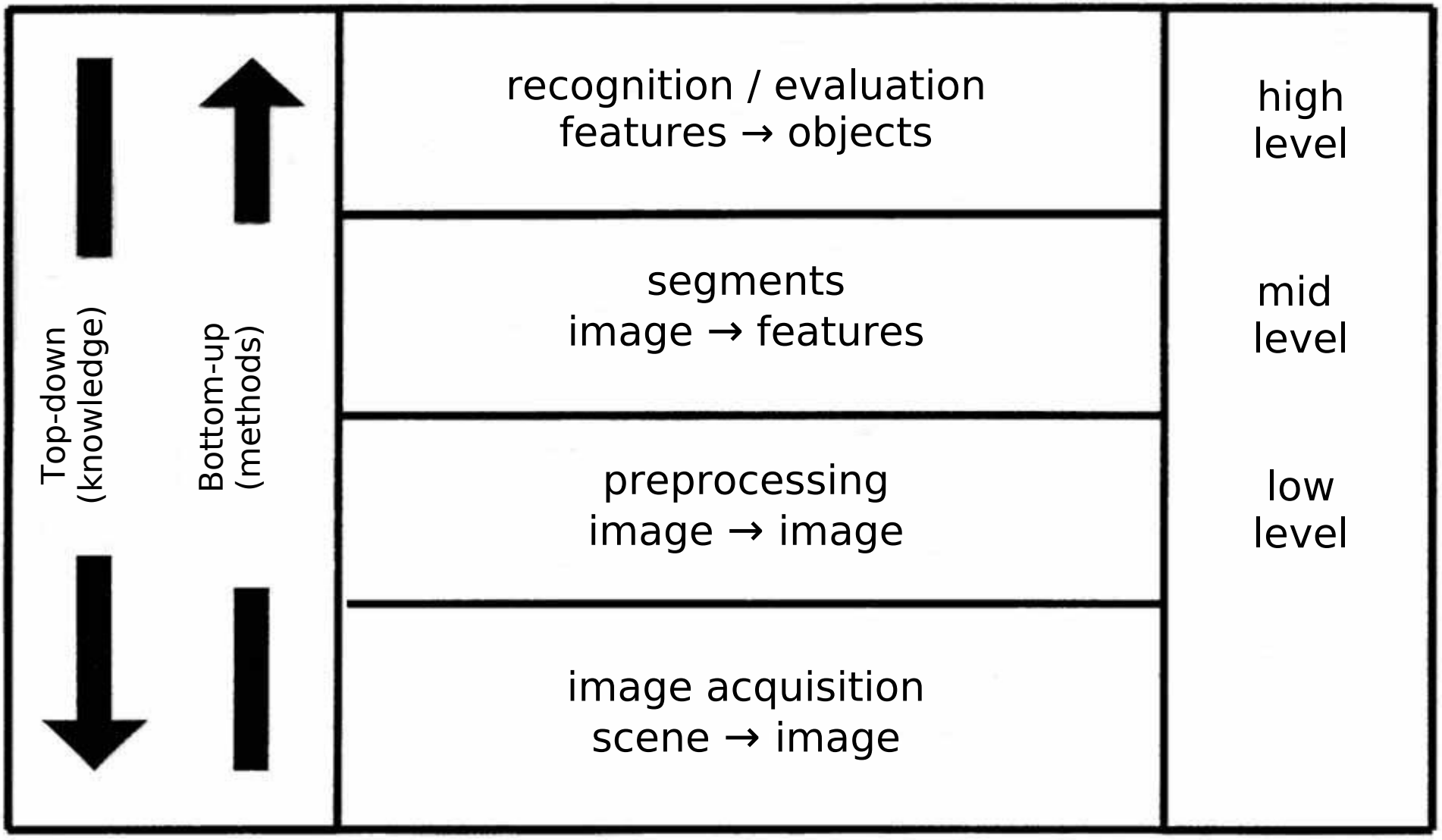
What are you gonna learn?

Photoshop, Gimp, …

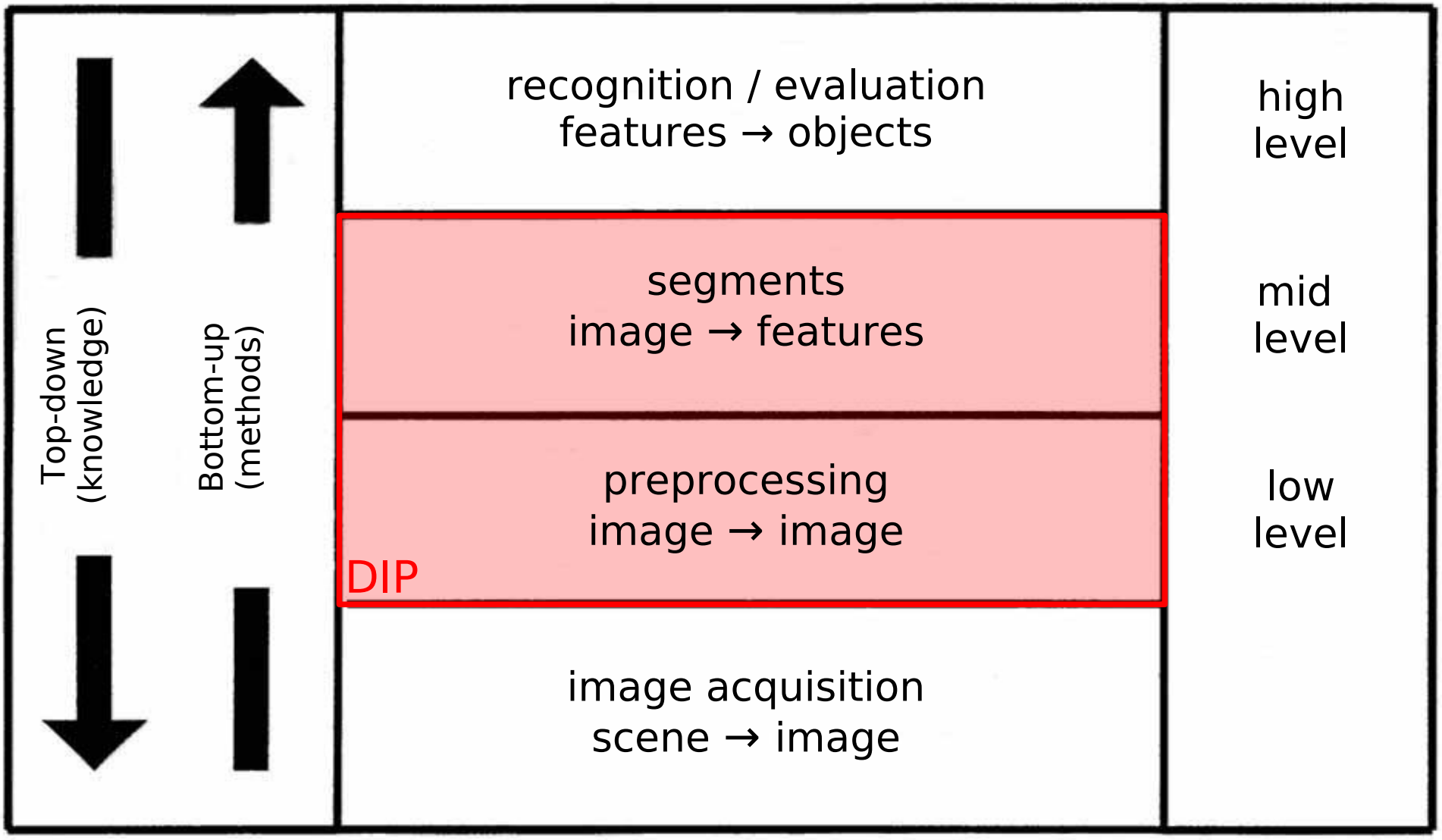


- NOT how to USE it (image editing)
- BUT how it WORKS (image processing)

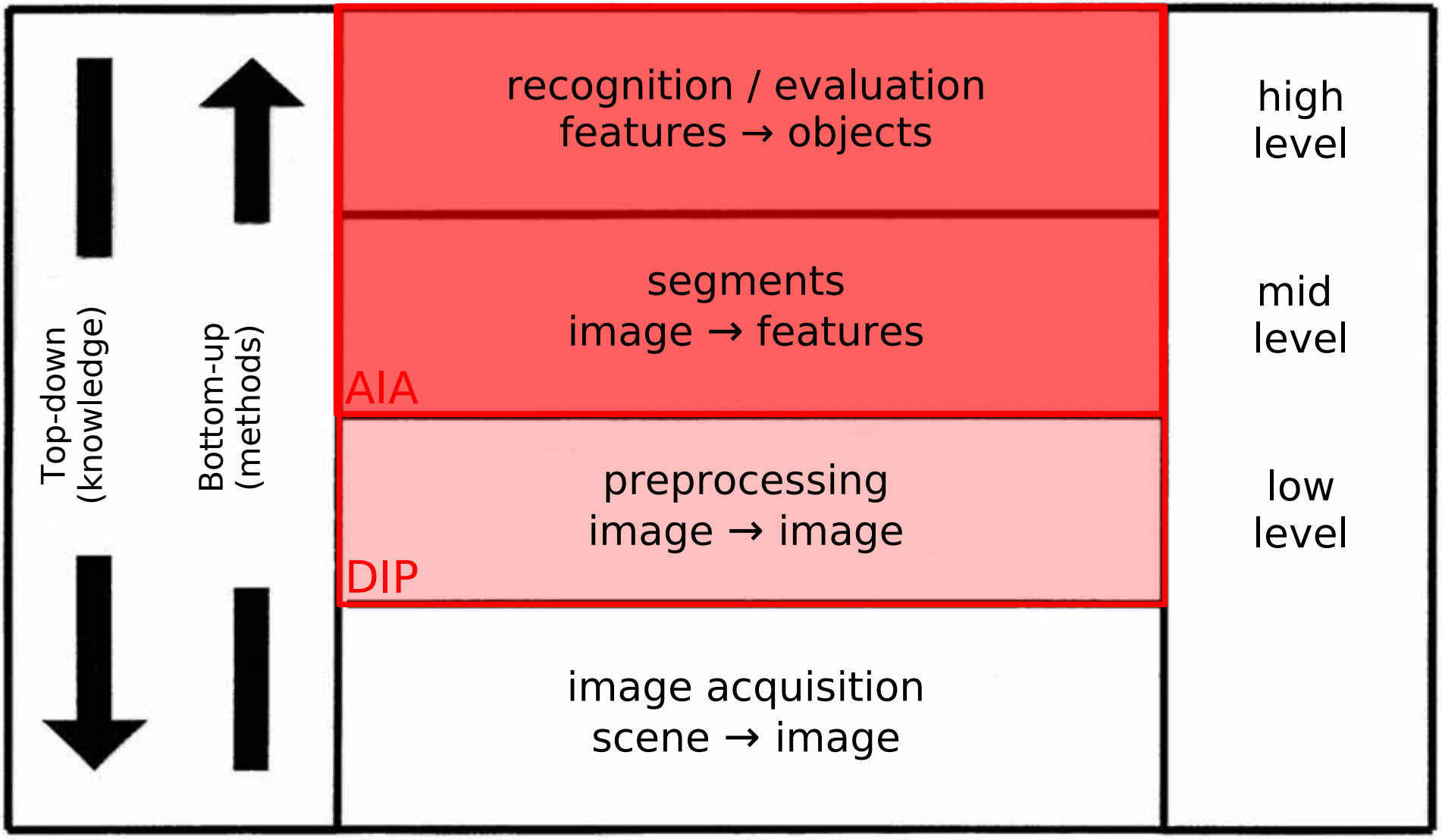
What are you gonna learn?



What are you gonna learn?



What are you gonna learn?



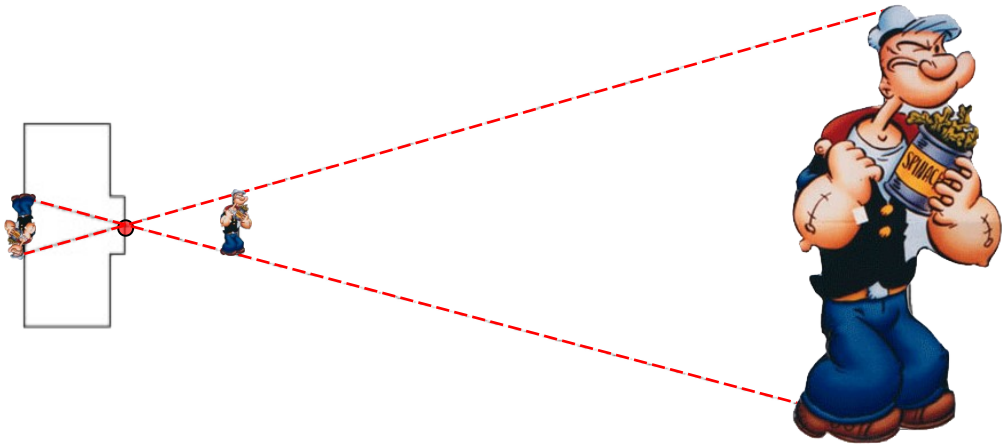
Overview

Motivation & Introduction

Overview

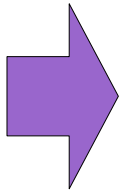
Motivation & Introduction

Image formation and geometric image transformations



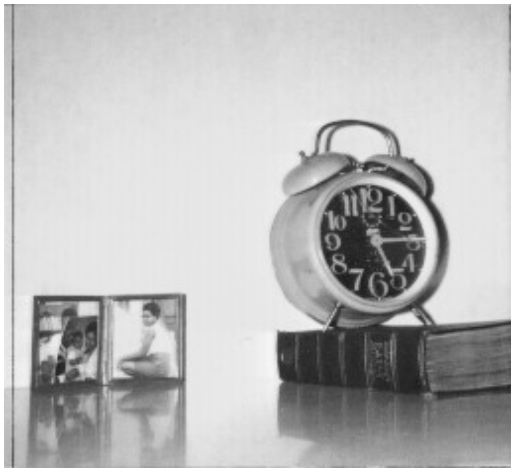
Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain



Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering



$f(x,y)$

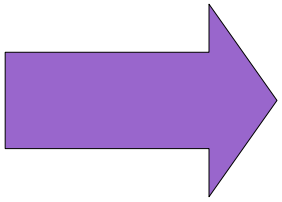
HPF

$|f * h|$

(colours inverted)

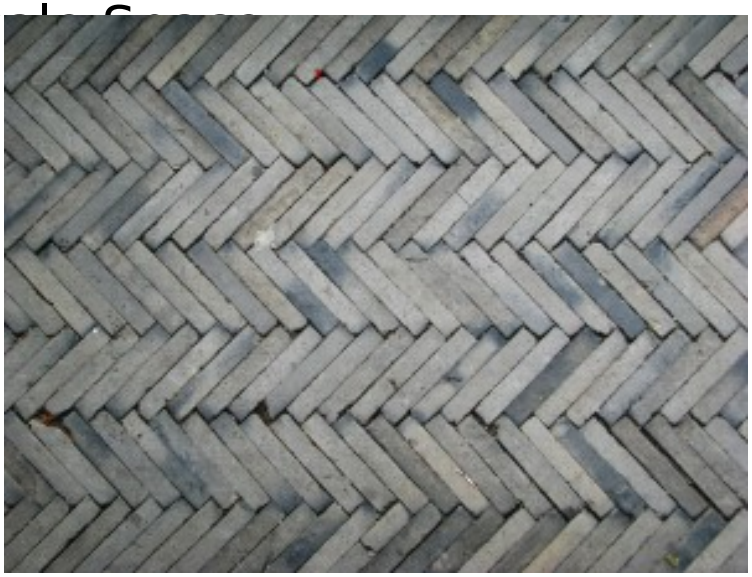
Overview

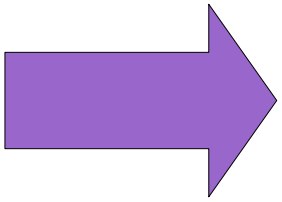
Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
T...



Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture





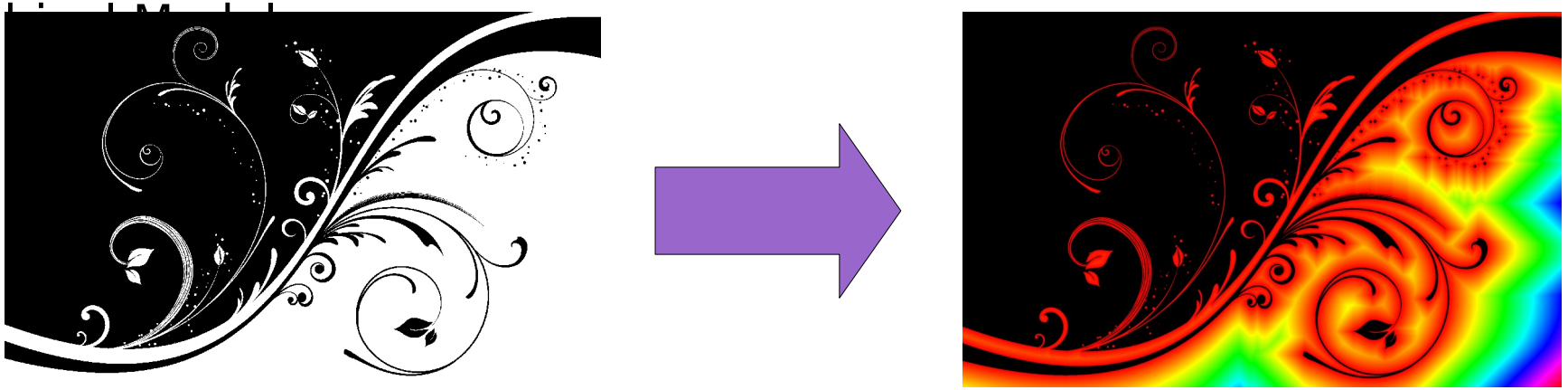
Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture
Scale Space



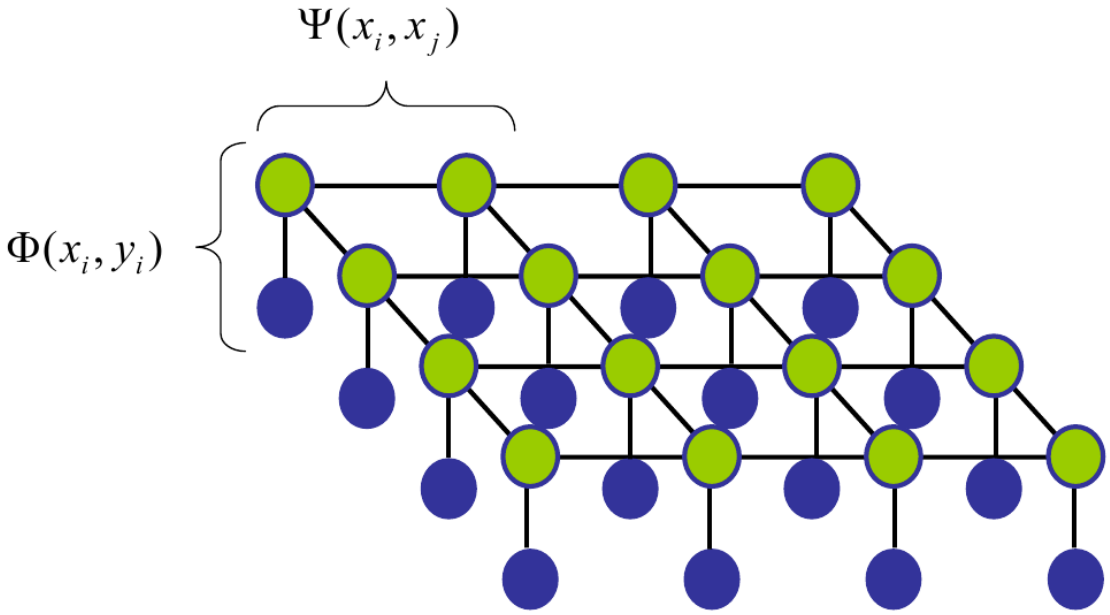
Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture
Scale Space
Mathematical Morphology



Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture
Scale Space
Mathematical Morphology
Graphical Models



Overview

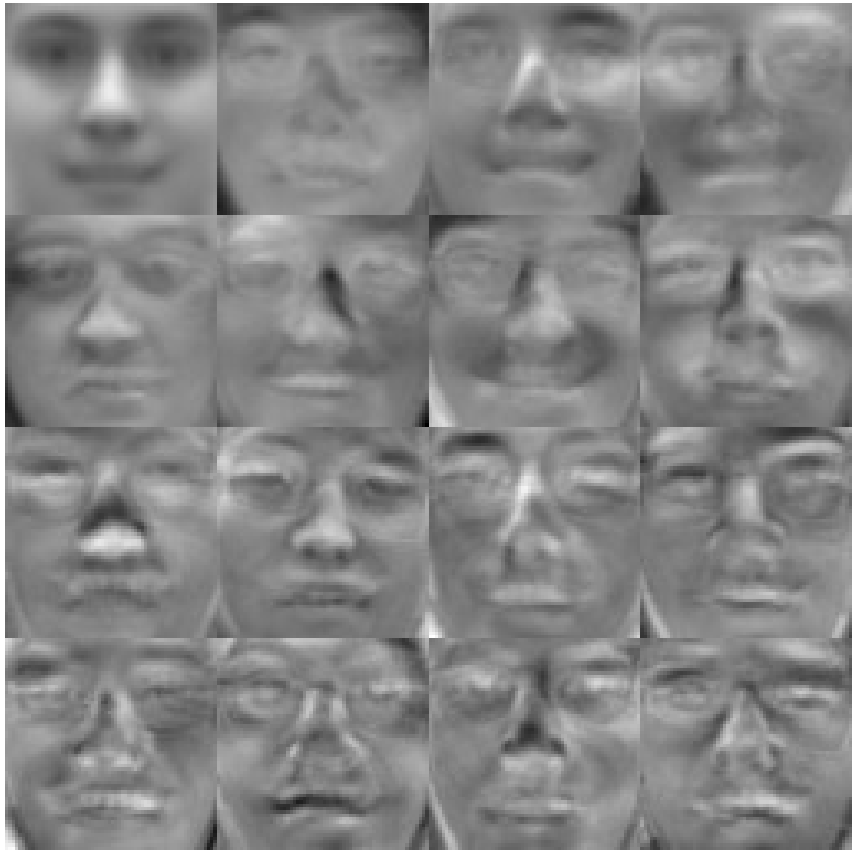
Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture
Scale Space
Mathematical Morphology
Graphical Models
Extraction of Salient Points

Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture
Scale Space
Mathematical Morphology
Graphical Models
Extraction of Salient Points
Extraction of Areas / Segmentation
Image Transformations

Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture
Scale Space
Mathematical Morphology
Graphical Models
Extraction of Salient Points
Extraction of Areas / Segmentation
Image Transformations / Representations



Overview

Motivation & Introduction
Image formation and geometric image transformations
Mask-/Filter Techniques
Convolution in Spatial Domain
Fourier Transform
Convolution via Frequency Domain
Smoothing / Low-Pass Filtering
Edge Extraction / High-Pass Filtering
Image Restoration
Texture
Scale Space
Mathematical Morphology
Graphical Models
Extraction of Salient Points
Extraction of Areas / Segmentation
Image Transformations / Representations

Outlook / Invited Talks

How are you gonna learn it?

1. Visit lectures
   - Every week (ER 164, Monday, 14-18 o'clock)

2. Visit exercises
   - Every two weeks (H 1028, Tuesday, 16-18 o'clock)

3. Doing homework
   - Consultation time: MAR6.043, Wednesday, 16:30-17:30 o'clock

4. ASK QUESTIONS!
   - Always! But: Ask me, not your neighbour

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)

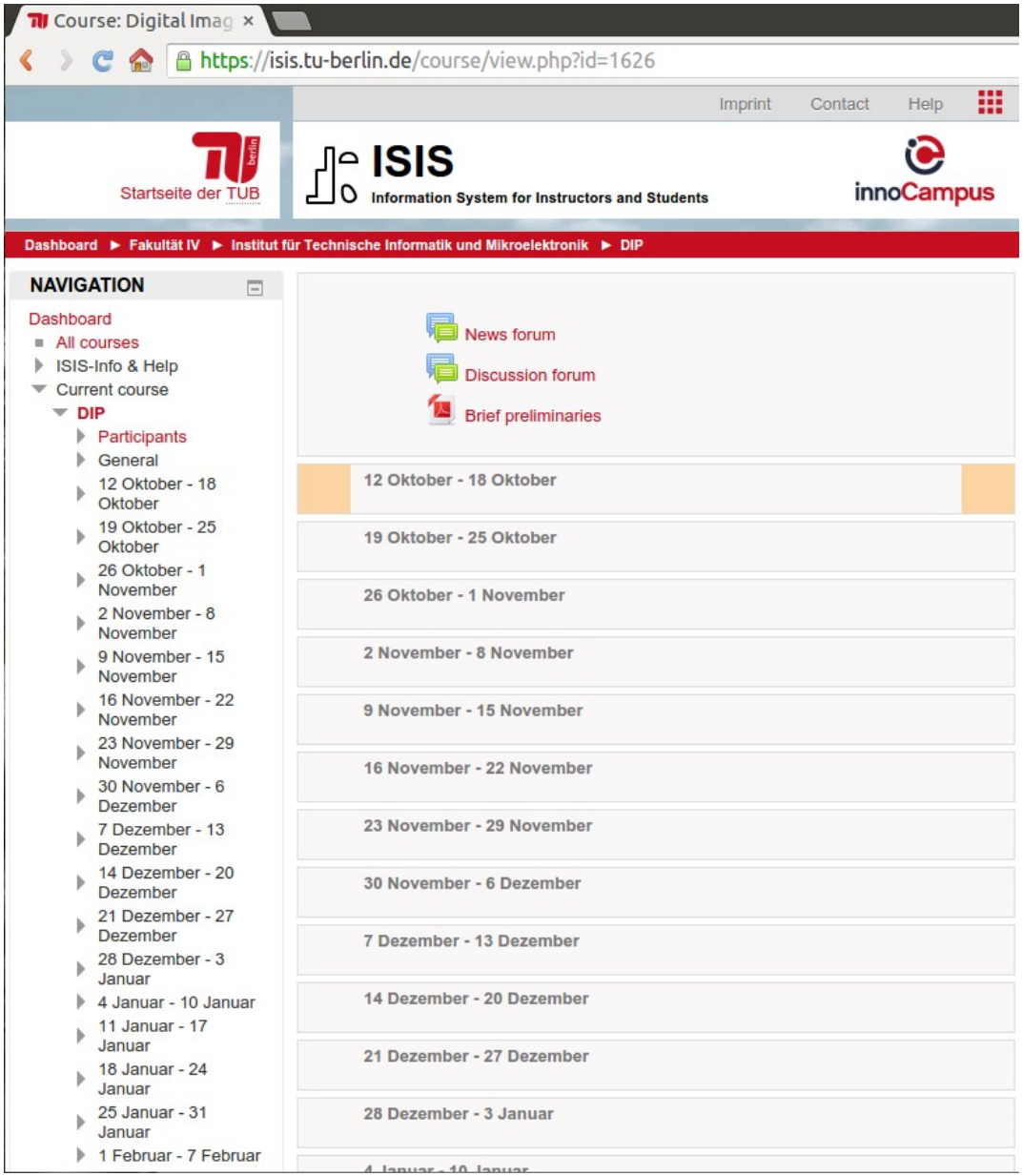
Informations

WWW

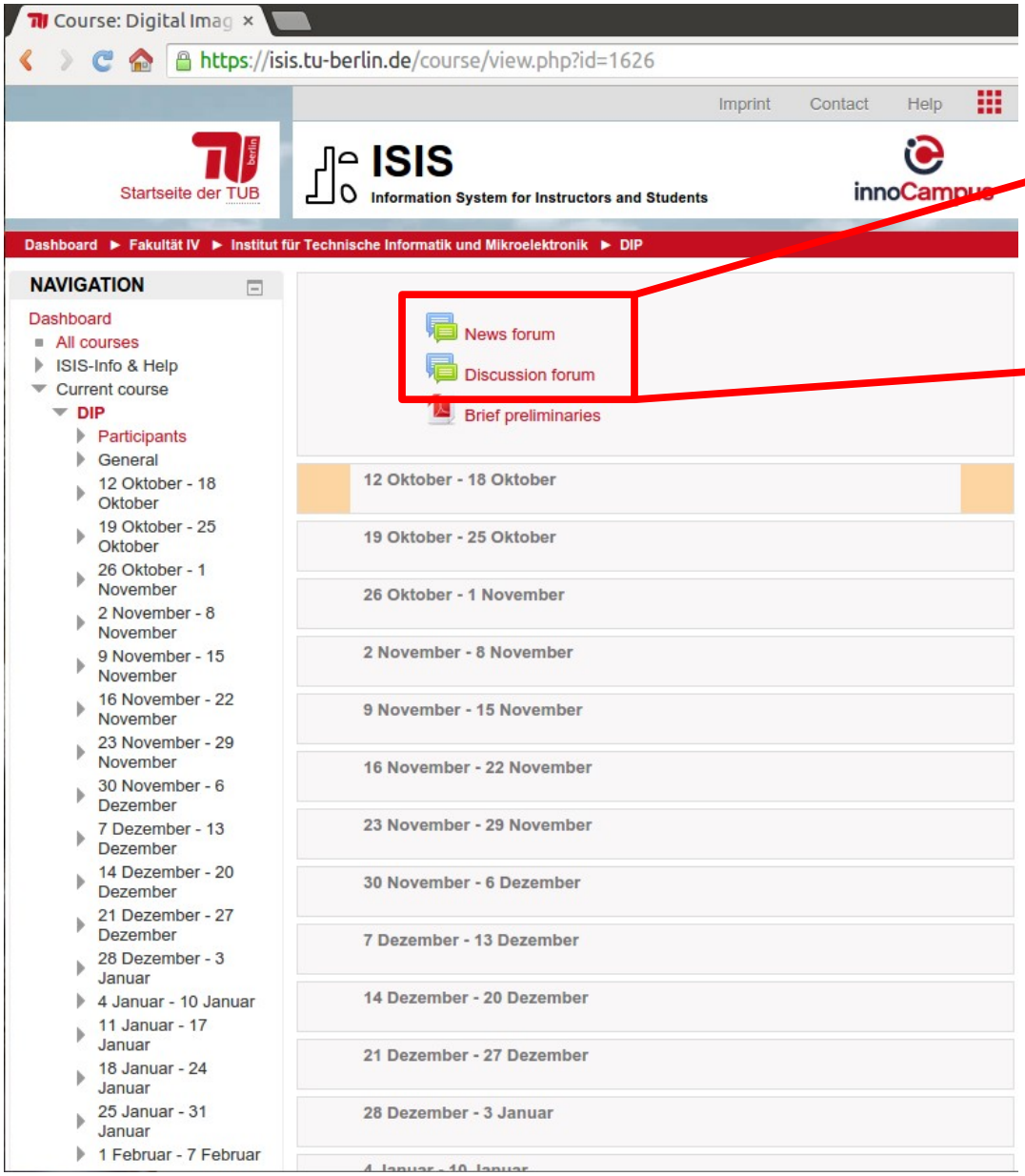
- Information, important announcements:
  - → http://www.cv.tu-berlin.de Announcements: 'Lectures'

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

Information



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 me! → EMail

Exams

- **Mid-term:**
  - Near the middle of the term: **10.12.2019**
  - Room: tba
  - Duration: 16 pm, 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: **17.02.2020**
  - Room: HE 101
  - Duration: 17-20 o'clock (90 min)

- Questions in English, answers are check marks (multiple choice)

Homework

What to do?

I: Answering theoretic questions

II: Implementation of methods for processing digital images

How?

Individual work (no group work)
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

- “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!

Homework

- **Theoretical Part:**
  - Questions in these slides
  - Answer in PDF form

- **Practical Part:**
  - Questions also in these slides
  - Download zip-file from isis
    - Contains bunch of source code files and possibly images
    - **Dip[1-6].cpp** contains empty/placeholder functions
    - **Fill in/complete those functions**
  - Algorithms more important than well-written code (but try!)

- **Submission:**
  - Due 2 weeks later
  - Via isis
  - One zip-file with: PDF, source code files, input and (if applicable) output images

Homework

- Next meeting in two weeks

- **BEFORE Tuesday, 16 o'clock**:
  - Hand in your solution via ISIS

  - Solution includes (red denotes **mandatory** material):
    - → **A single .zip file (no rar, 7z, ...) containing**
    - → **No subdirectories** (unless in original zip file)
    - → **All files** of the **provided material**
      - i.e. all .h and .cpp files, CMakeLists.txt
      - But no executables, object files, stdafx.h, …!
    - → **Input** and **output images** (if applicable)
    - → **Pdf-file with**
      - → **Student name, and student ID**
      - → Short discussion / presentation of your solution
      - → **Answers to theoretical questions**

Homework

Don’t break my automated tests:

- Do not modify:
  → headers, behaviour of given functions, file names, tests
- Only modify the Dip[1-6].cpp file
  - (Unless specified otherwise)
- Do not rename files, use subdirectories in zip file, use a format other than .zip, ...

It has to work on my machine:

- Code that doesn't compile won't be checked
- Do not use absolute paths to headers, images, and other files
- If unsure, check with VM

Plagiarism is considered a failure:

- Submitting solutions from previous years is plagiarism too
- You immediately fail the homework criterion

Plagiarism

- **We will look (maybe sporadically) for plagiarism**
- **If we find plagiarism in your homework submissions:**
  - **You will immediately fail the homework criterion and can't take the exam!**
  - **No second chances!**
  - **I don't care that you [insert lame excuse here]**

Plagiarism

- “I didn’t have the time”
- “I can’t do it myself”
- “I accidentally send the wrong file”
- “I could do it myself, but my english is not as good as that on wikipedia”
- “Where I’m from, even the professors do it”
- “I can totally explain how the other student’s name is also on my submission”
- “But it’s my work, the other student [1 year ago built a time machine and] copied from me”

→ Trust us, we have heard it all before

Plagiarism

- “But I didn’t know about it”
  - I just told you
  - Even if you truly didn’t know:
    - Courts ruled (multiple times) that it is still plagiarism and enforceable.
- “Ok, you caught me, but you didn’t catch others. That is unfair treatment!”
  - Courts usually rule that it doesn’t matter. The others just got lucky.

See e.g. https://kops.uni-konstanz.de/handle/123456789/37223

So what is Plagiarism?

As a rule of thumb:

- DO NOT COPY!
- Your submission must be your own, unique solution.
- Just do your homework yourself and you'll be fine.

- It's plagiarism if you copy from
  - The internet (even if it's google translated from another language)
  - Other students (same or past years)
- It's also plagiarism if you let s.b. else do your homework.

So what is Plagiarism?

- It’s plagiarism even if you didn’t ctrl+c/v but retyped it.
- It’s still plagiarism if you only changed individual words/variable names or only made other marginal changes.
- Letting others help you is ok. But:
  - ➔ If “help” means they dictate what to type, it’s again plagiarism.

Plagiarism

If we find any of those, or s.th. in the same spirit, you will immediately fail the homework criterion and can't take the exam!

Programming Environment

Programming Environment

You will need a couple of tools and libraries installed. Two options:

Install on your OS

You'll need:

- C++ compiler
  - → gcc/llvm/msvc
- cmake (optionally)
- OpenCV library
- Eigen library (PCV only)
- IDE (recommended)
  - → Code::Blocks/xCode/ Visual Studio

- Usually easy on linux, hard on windows

or

Use our virtual machine

- Install VirtualBox
- Download our VM image
- Contains:
  - → Linux
  - → All libraries and tools preinstalled

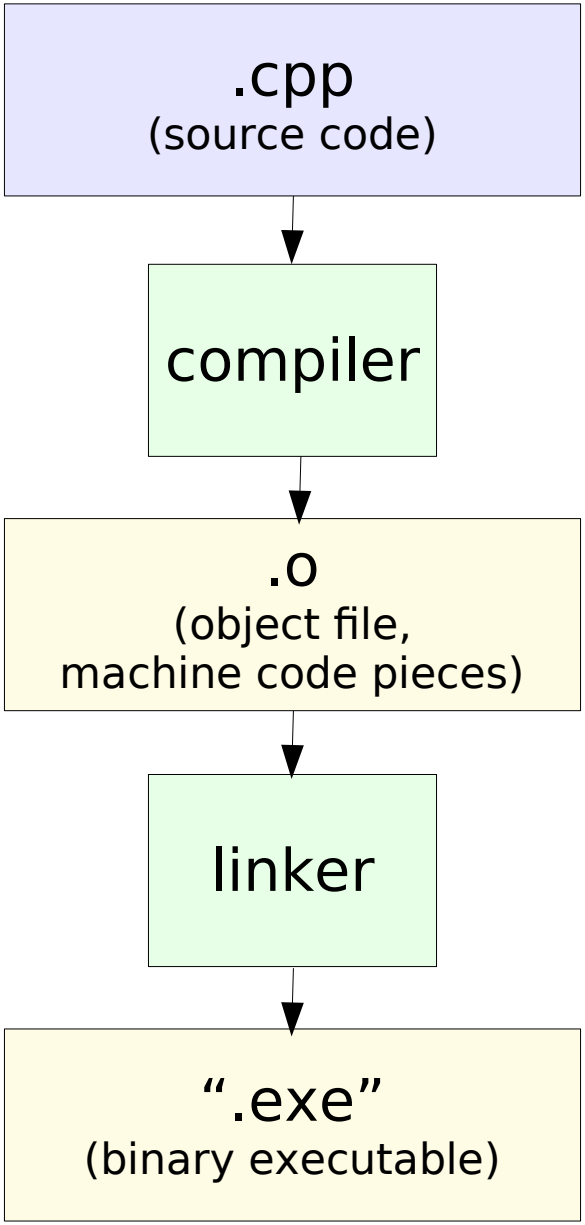
- Also serves as a reference
  - → Code has to work here

Virtual Machine Setup

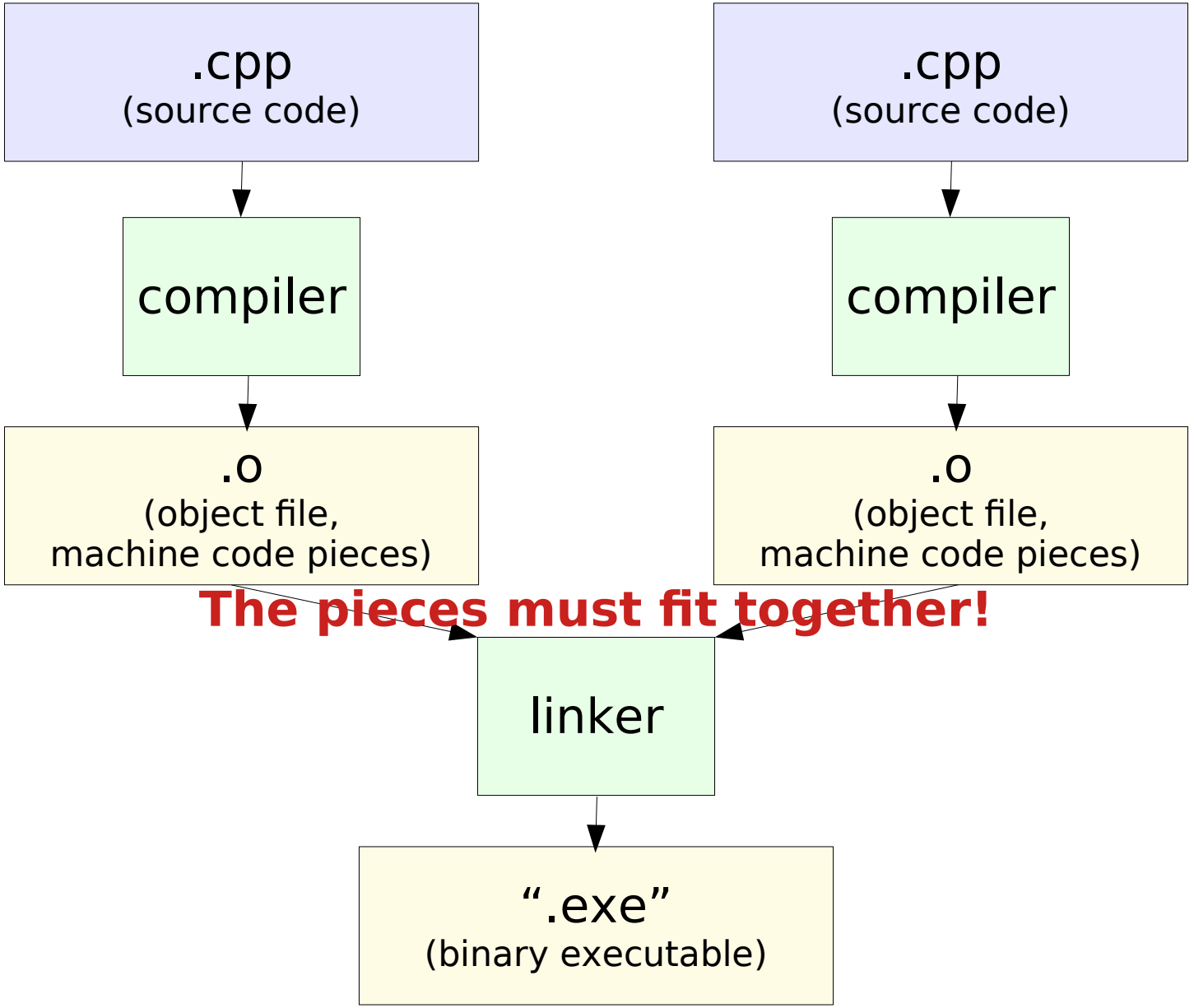
- Download from here (.ova image file)
- Install VirtualBox
- Open main Virtual Box window “Oracle VM VirtualBox Manager”
- Select “File” → “Import Appliance…”
- Select .ova image
- Tick “Reinitialize the MAC address of all network cards”
- Hit “Import”
- After import, select the “CVTeachingMachine” from the list of VMs and hit “Start”

Building process

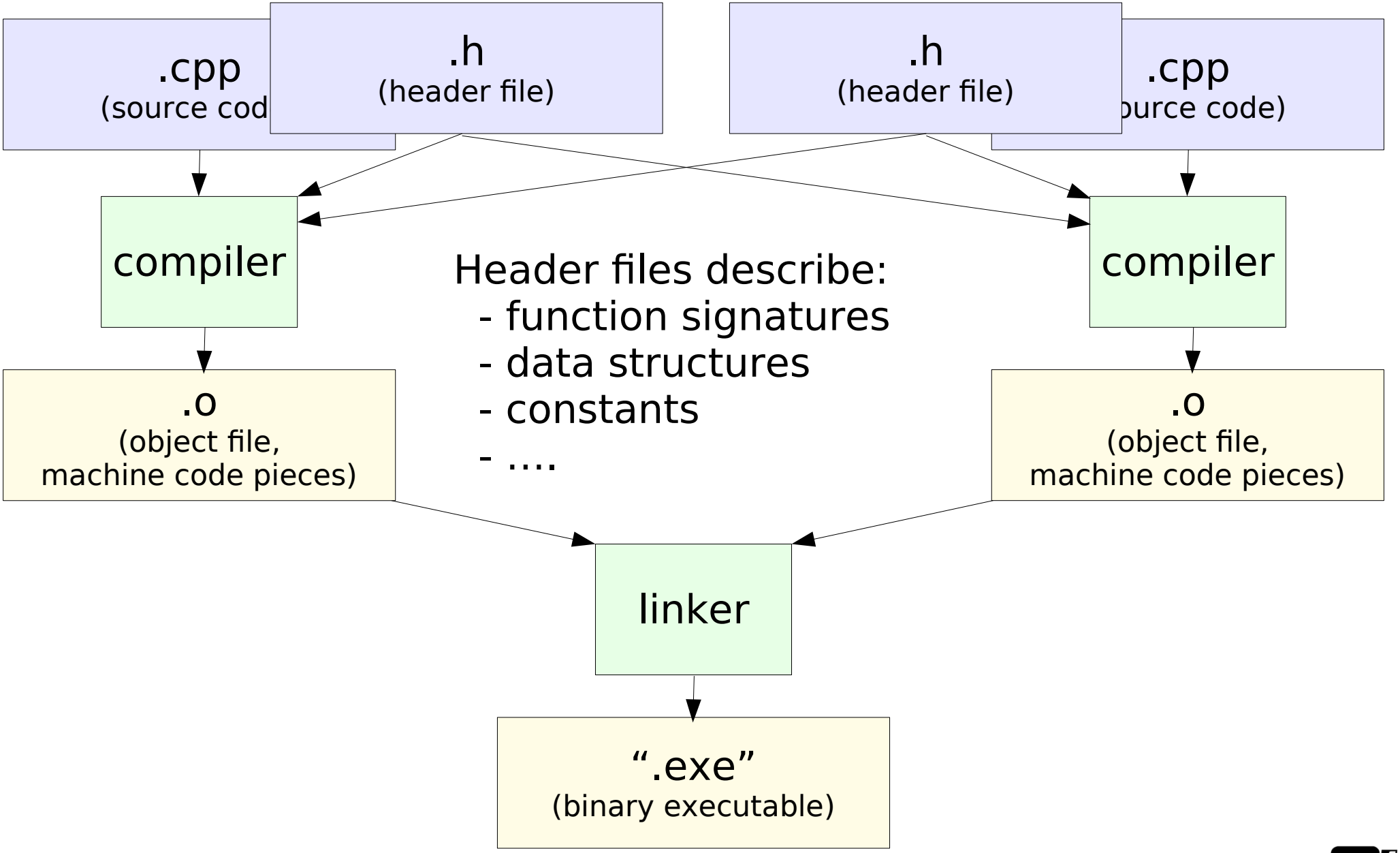
C++ - Building Programs



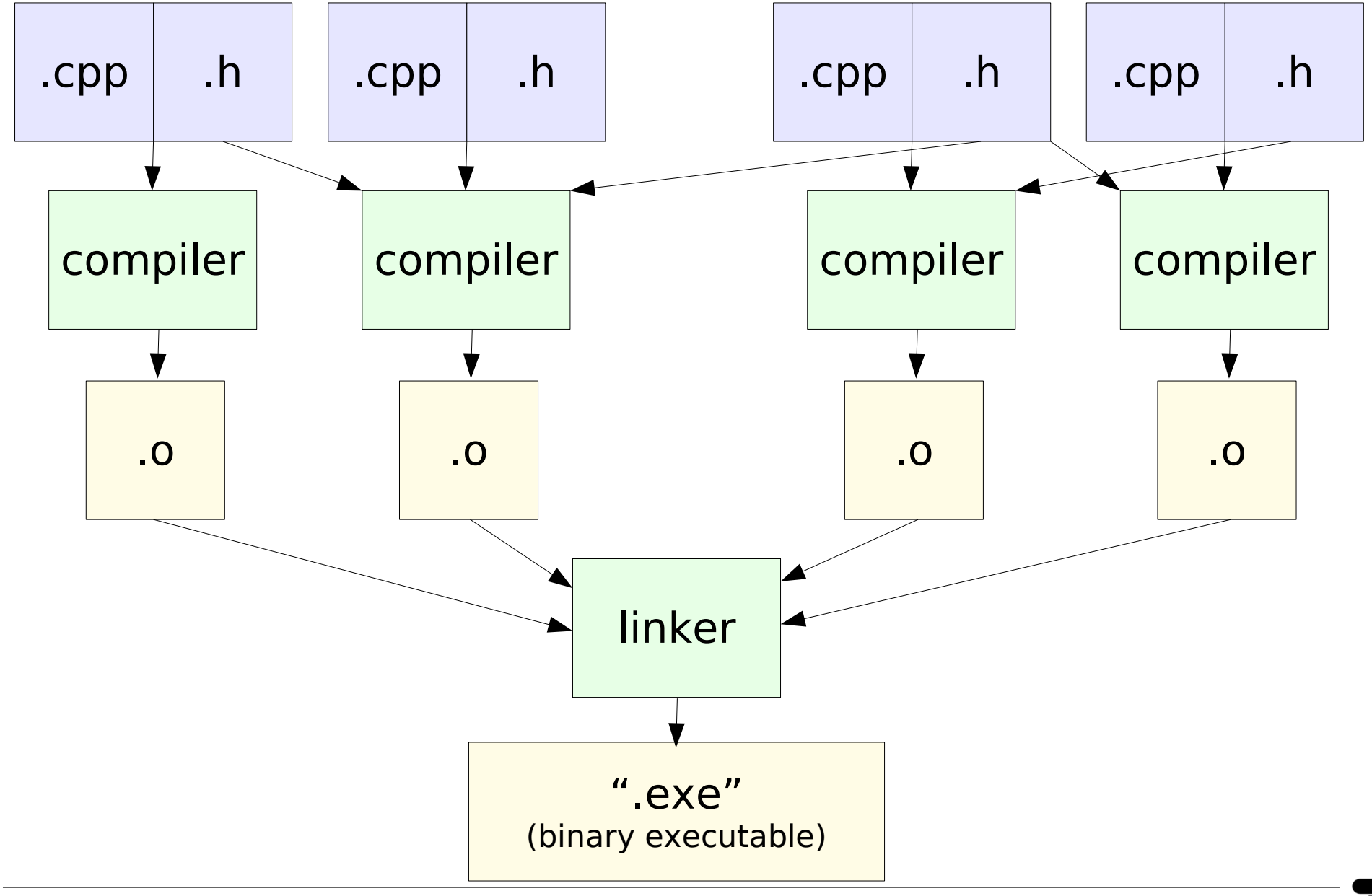
C++ - Building Programs



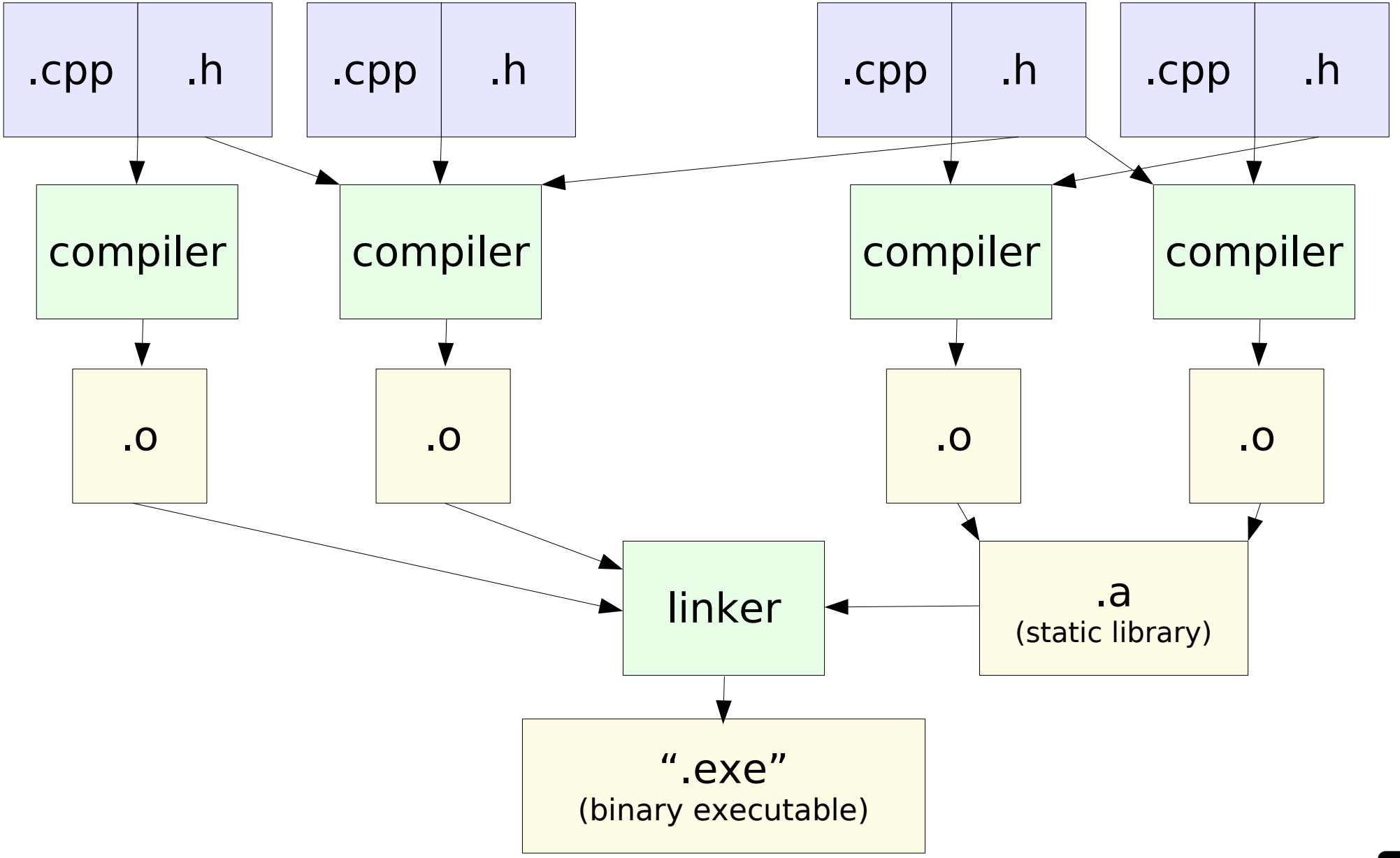
C++ - Building Programs



C++ - Building Programs



C++ - Building Programs



CMAKE

We use CMAKE

- *CMake* builds project files for common build systems
- Some IDEs can directly open *CMake* files

Example on linux with *make*:
Create out-of-source build directory and setup debug build:

```
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Debug
```

Once setup, build with *make*:

```
make -j
```

Use “-DCMAKE_BUILD_TYPE=Release” for faster release build.

C++ - Tools

- C++ programs can be build with only command line tools
  - e.g., gcc, clang

- Integrated Development Environment (IDE) is strongly recommended
  - Syntax Highlighting
  - Integrated build system
  - Debugger
  - Auto-Complete
  - ...

    - Linux:
      - → Kdevelop
    - Mac OS:
      - → Xcode
    - Windows:
      - → Visual Studio
    - Cross-Platform:
      - → Code::Blocks, Eclipse, QtCreator, Visual Studio Code, ...

CMAKE + Code::Blocks

Example on linux with Code::Blocks:
Create out-of-source build directory and setup debug build project structure for Code::Blocks:

```
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Debug \
    -G “CodeBlocks - Unix Makefiles”
```

Once setup, open project in Code::Blocks.

This is the recommended way in the VM!

CMAKE Executables

Cmake is configured to build two executables:

./main:
- The actual “demo” program that showcases the algorithm

./test:
- Runs small tests to check correctness of implementation
- These tests **must** succeed for homework to pass as correct

These two executables are exposed in some IDEs as different build/debug targets:
- For debugging, select which one you want to run

Also remember to set program arguments (e.g. path to test images)
- Code::Blocks: Project → Set program’s arguments...

Syntax and Pitfalls

C++ for Programmers

- This is not a course about programming
- If you can’t code (yet):
  - Tons of tutorials and videos on the internet
- If you can code (python, java, js, matlab, …)
  - You should be fine, even without c++ skills
  - Most homework is “fill in the gaps”-style
  - But: C++ has a couple of non-trivial quirks
    - Let’s look at those

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

- Which .h files to include
- Path relative to file or search paths given to compiler
- Linker still needs to link against lib

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

- Function
- No parameters
- Return type int
- “main” function gets called when program is executed

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

- Scopes / Code blocks with { }
- Indentation is ignored
- Multiple instructions in one line possible

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

- Semicolon ‘;’ after every instruction mandatory

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

- Namespaces prevent identifiers from clashing
- Namespace name forms a prefix with a double colon
- “using namespace std;” (with std/cv/...) gives default access
  - ➔ Only do this in .cpp files

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

- String literals must be delimited by double quotation marks (“ “)
- Single quotation marks (‘ ‘) for single characters only

C++ - Syntax

C++

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;

    cv::Mat image = cv::imread("image.png");

    cv::imshow("img", image);

    cv::waitKey(0);

    return 0;
}
```

Python

```python
import cv2



def main():

    print("Hello World")

    image = cv2.imread("image.png")

    cv2.imshow("img", image)

    cv2.waitKey(0)

if __name__ == "__main__":
    main()
```

- C++ is strongly typed
- All variables must have known type at compilation time
- Type must be specified when variable is declared
  - → Same holds for function parameters
  - → And function return values

C++ - Types

C++

```cpp
float sqrt(float x, unsigned iters) {
    float result = x / 2;
    for (unsigned i = 0; i < iters; i++)
        result = (result + x / result) / 2;
    return result;
}
```

Python

```python
def sqrt(x, iters):
    result = x / 2
    for i in range(0, iters):
        result = (result + x / result) / 2
    return result
```

- Types can be basic types (int, float, ...) or more complex data structures (e.g. classes)
- Compiler checks compatibility (at compilation time)

Some important types:

bool	Boolean value	true, false
int	Signed integer number	-1, 2, 1337
unsigned	Non-negative integer number	0u, 1u, 12345u
char	Single character	’a’, ‘b’, ‘C’
float	32-bit floating point number	1.0f, 4.0f, 1e-2f
double	64-bit floating point number	1.0, 2.0, 3e-4

C++ - Types

C++

```cpp
float sqrt(float x, unsigned iters) {
    float result = x / 2;
    for (unsigned i = 0; i < iters; i++)
        result = (result + x / result) / 2;
    return result;
}
```

Python

```python
def sqrt(x, iters):
    result = x / 2
    for i in range(0, iters):
        result = (result + x / result) / 2
    return result
```

- Types can be basic types (int, float, ...) or more complex data structures (e.g. classes)
- Compiler checks compatibility (at compilation time)

Some important classes:

std::string	String (text)	#include <string>
std::vector<type>	Dynamic array. Type of elements must be specified inside < >	#include <vector>
std::list<type>	Linked list	#include <list>
std::unique_ptr<type>	Smart pointer (more later)	#include <memory>
std::shared_ptr<type>	Smart pointer (more later)	#include <memory>

C++ - Types

- Implicit type casts:
  - “1+2” → “3” (int)
  - “1+2.5f” → “3.5f” (float)
  - “1 / 2.0” → “0.5” (double)
- Some pitfalls:
  - “10 / 4” → “2” (int)
  - “0u – 1u” → “4294967295” (unsigned) (hardware dependent)
  - “2e9f + 1” → “2e9f” (float, unchanged)

C++ - Operators

The usual suspects	a + b, a – b, a / b, a * b
Modulo	a % b
Bitshift by b left, right	a << b, a >> b
Increment / decrement	a++, a--
Assignment	a = b
Compute and assign	a += b, a -= b, a *= b, a /= b, a %= b
Type cast	(float) b
Force float division	a / (float) b

C++ - Boolean Expressions

Equal	a == b
Not equal	a != b
Greater, lesser	a > b, a < b
Greater equal, lesser equal	a >= b, a <= b
Assign b to a and check if unequal to zero	a = b

C++ - Boolean Operators

For booleans

For all bits in an integer

Operation	Logical	Bitwise
not	!a	~a
and	a && b	a & b
or	a || b	a | b
xor	a != b	a ^ b

C++ - Operators

Operators can be overloaded by classes:

- printing in STL

```
10          std::cout
11              << bottles << " bottles of beer on the wall," << std::endl
12              << bottles << " bottles of beer." << std::endl
13              << "Take one down, pass it around," << std::endl;
```

- vector math in Eigen

```
10      Eigen::Matrix3f A = ...;
11      Eigen::Vector3f b = ...;
12
13      Eigen::Vector3f c = A * b;
```

C++ - Flow Control

Boolean expression must be in parantheses

```
10      if (a == b) {
11          DoSomething_A();
12          DoSomething_B();
13          DoSomething_C();
14      }
```

Multiple conditional instructions scoped into { }

Single conditional instruction can be without { }

```
10      if (a == b)
11          DoSomething_A();
12      DoSomething_B();
13      DoSomething_C();
```

These are equivalent!

```
10      if (a == b)
11          DoSomething_A();
12          DoSomething_B();
13          DoSomething_C();
```

Python users be warned:
Indentation is ignored!

C++ - Flow Control

Boolean expression must be in parantheses

```
10      if (a == b) {
11          DoSomething_A();
12          DoSomething_B();
13          DoSomething_C();
14      }
```

Multiple conditional instructions scoped into { }

Single conditional instruction can be without { }

```
10      if (a == b)
11          DoSomething_A();
12      DoSomething_B();
13      DoSomething_C();
```

Counts as “no conditional instructions”

```
10      if (authorized);
11          reveal_secret();
```

Python users be warned:
Indentation is ignored!

C++ - While Loops

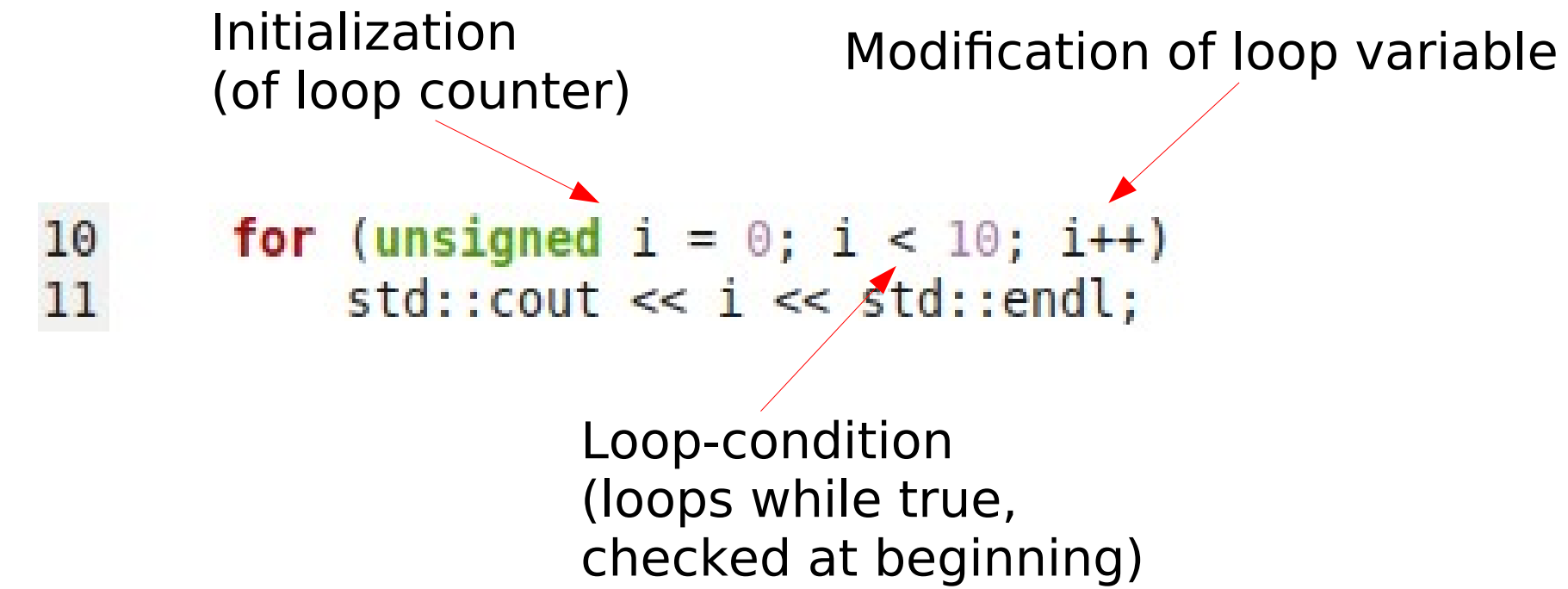
- “While” and “do-while” loops very similar to “if“
- Same indentation pitfall for python users!

```
10    while (!done) {
11        do_awesome_stuff();
12        display_useless_information();
13    }
```

```
10    do {
11        do_awesome_stuff();
12        display_useless_information();
13    } while (!done);
```

C++ - For Loops

Classical for loop with three part header:

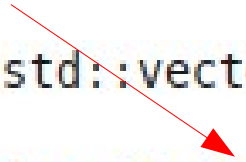


C++ - Ranged For

Shorthand to iterate over containers

```
10      std::vector<int> coolNumbers = { 42, 1337, 31337};
11
12      for (int i : coolNumbers)
13          std::cout << i << std::endl;
```

If modifications are to be made:



```
10      std::vector<int> coolNumbers = { 42, 1337, 31337};
11
12      for (int &i : coolNumbers)
13          i *= 2;
```

C++ - continue/break

- In all types of loops:
  - “break;”
    - immediately jumps out of the loop
  - “continue;”
    - skips the rest of this loop iteration
    - continues with next loop iteration

C++ - Functions

Return type
(“void” for none)

Function name

Parameters with types

```
10 int computeAnswerToUltimateQuestion(float lots, unsigned of, const std::string &data)
11 {
12     std::this_thread::sleep_for(std::chrono::years(7'500'000'000));
13     return 42;
14 }
```

Specify return value (also skips rest of function)

Pass complex objects as “const reference” to prevent copy

C++ - Value vs Reference

```
10    int a = 0;
11    int b = a;
12    b = 42;
13    // a still 0
```

“b” is just a copy of a

```
10    int a = 0;
11    int &b = a;
12    b = 42;
13    // a also 42
```

“b” is a reference to a (alias of the same instance)

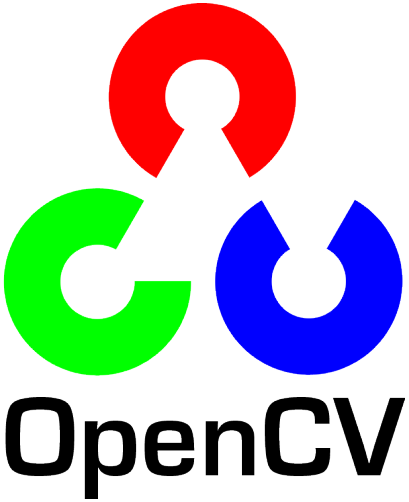
Function parameters work exactly the same!
(Useful for returning more than one result)

```
10 void func(int b) {
11     b = 42;
12 }
13
14 // ...
15     int a = 0;
16     func(a);
17     // a still 0
```

```
10 void func(int &b) {
11     b = 42;
12 }
13
14 // ...
15     int a = 0;
16     func(a);
17     // a also 42
```

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.... 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);
}
```

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

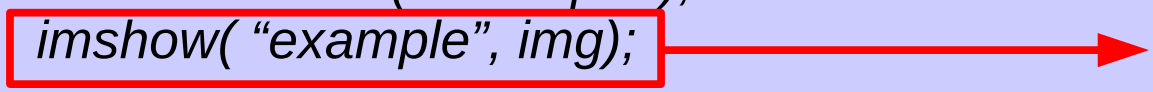
Brief introduction to OpenCV

```
#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;
}
```

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)`

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:

- CMake based build system (CMakeLists.txt)
- Main function (main.cpp)
- Test function (test.cpp)
- Function declaration (Dip1.h)
- Basic functionality (Dip1.cpp)

Todo:

- **[Setup VM]**
- Optionally: **[Install programming environment directly]**
- Dip1.cpp
  - Mat Dip1::doSomethingThatMyTutorIsGonnaLike(Mat&)
    → Do something (reasonable)

Deadline:

- Next meeting at **29.10.2019**, 16pm

1. Exercise – Part II : Given

FILE: main.cpp
int main(int argc, char argv)**

- Main function
  - Usage:
    - ./main path_to_image
      - Calls dip1::run(...)
      - Which calls dip1::doSomethingThatMyTutorIsGonnaLike(..)

FILE: test.cpp
int main(int argc, char argv)**

- Main function
  - Usage:
    - ./test path_to_image
      - Calls test_doSomethingThatMyTutorIsGonnaLike(...)
      - Which calls dip1::doSomethingThatMyTutorIsGonnaLike(..)

1. Exercise – Part II : To Do

```
Mat doSomethingThatMyTutorIsGonnaLike(Mat& img)

img      :   input image
return   :   output image
```

→ does something cool... (hopefully)