Computer Vision Exercises

Simone Gaffling
14.04.2014
Pattern Recognition Lab (CS 5)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- Organizational Matters
- Outline of the Exercises
- Introduction to OpenCV
- Image Representation
- Image Manipulation

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Organizational Matters (1/3)

- Exercises are always on Monday and Wednesday
- Attending the exercises is mandatory
- Sending in the exercises is voluntary
  BUT: if you need feedback on your work, send your solutions/questions!

- New exercises will be uploaded each Monday morning

- 1 SWS = 45 minutes
- Due to public holidays: 2 Monday sessions missing
  ➔ will be covered the following week (→ 90 min exercise)

Organizational Matters (2/3)

- Contact
  - simone.gaffling@cs.fau.de
  - room 09.132 (opposite site)
  - Office-days: mon, wed, fri (till noon)
- Exercises, Slides and Data
  - http://www5.cs.fau.de → Courses → SS 14 → Computer Vision Exercises [CV-E]
  - Will be still available after this term

www.ideal-conferences.net

Outline of the Exercises

- How to apply computer vision techniques?
- Further information on theoretical background

- OpenCV is used for algorithms
- Programming should be done in C++
- Please comment your code extensively

```
//for oscillation testing
bool down = false;
bool up = false;

//flag for whether the correct threshhold has been reached
bool thresh_good = false;

Ptr<AdjusterAdapter> adjuster = adjuster_->clone();

//break if the desired number hasn't been reached.
int iter_count = escape_iters_;

while( iter_count > 0 && !(down && up) && !thresh_good && adjuster->good() )
{
    keypoints.clear();

    //the adjuster takes care of calling the detector with updated parameters
    adjuster->detect(image, keypoints,mask);

    if( int(keypoints.size()) < min_features_ )
    {
        down = true;
        adjuster->tooFew(min_features_, (int)keypoints.size());
    }
    else if( int(keypoints.size()) > max_features_ )
    {
        up = true;
        adjuster->tooMany(max_features_, (int)keypoints.size());
    }
    else
        thresh_good = true;

    iter_count--;
}
```

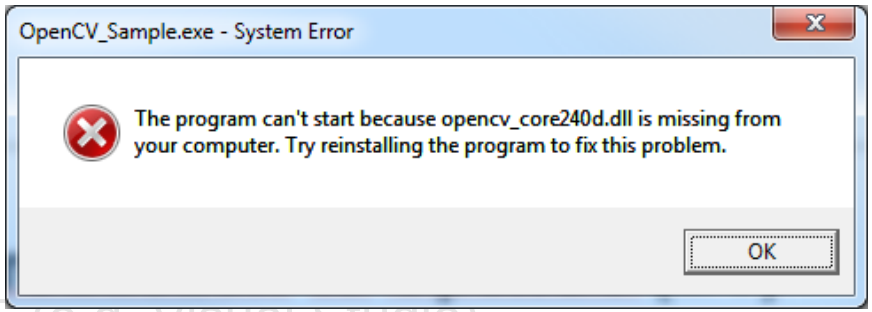
Introduction to OpenCV (1/3)

- **Open** Source **C**omputer **V**ision Library
- Computer vision and machine learning
- BSD lincense (everything allowed, but keep © info)
- More than 2500 algorithms:
  - Recognize faces
  - Identify objects
  - Track camera movements
  - Stereo vision
  - …
- C++, C, Python and Java interfaces
- Windows, Linux, Android and Mac OS
- Developed in C++ (CUDA support)

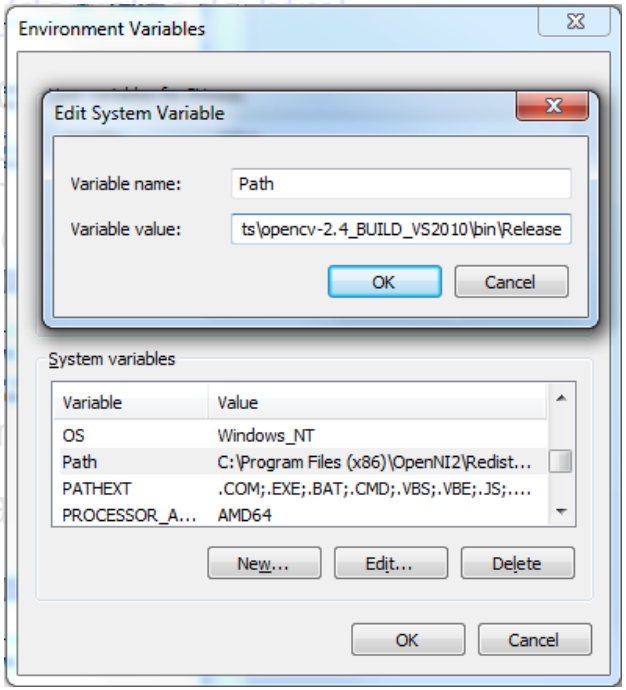
Introduction to OpenCV (2/3)

- Install any IDE (e.g. Visual Studio Express)
- Download latest OpenCV version
  - http://opencv.org/downloads.html
- Compile OpenCV for your system (CMake)
- Download sample project
  - [CV-E] homepage → „OpenCV_Sample.zip“
    - Unzip
    - CMake
    - OpenCV_Sample „Set as StartUp Project“
- Compile and run sample code

Introduction



- Install any IDE (e.g. Visual Studio)
- Download latest OpenCV versi
  - http://opencv.org/
- Compile OpenCV fo
- Download sample p
  - [CV-E] homepag
    - Unzip
    - Cmake
    - OpenCV_Sam
- Compile and run sa



Introduction to OpenCV (3/3)

-
```
// main.cpp : Defines the entry point for the console application.

#include <cv.h>
#include <cxcore.h>
#include <highgui.h>

int main(int argc, char* argv[])
{
        cv::Mat img = cv::imread("lena.jpg");

        cv::namedWindow("Lena", 1);
        cv::imshow("Lena", img);

        cv::waitKey();

        return 0;
}
```

Introduction to OpenCV (3/3)

```cpp
// main.cpp : Defines the entry point for the console application.

#include <cv.h>
#include <cxcore.h>
#include <highgui.h>

int main(int argc, char* argv[])
{
        cv::Mat img = cv::imread("lena.jpg");

        cv::namedWindow("Lena", 1);
        cv::imshow("Lena", img);

        cv::waitKey();

        return 0;
}
```

- Include OpenCV headers
- Use cv::Mat for images

Image Representation

- What is photography?

Image Representation

- What is photography?

- Wiki: phot → light and graphos → drawing
  - Record light or any electromagnetic radiation
  - Chemically on a light sensitive material
  - **Electronically on an imaging sensor**

Image Representation

- What is photography?

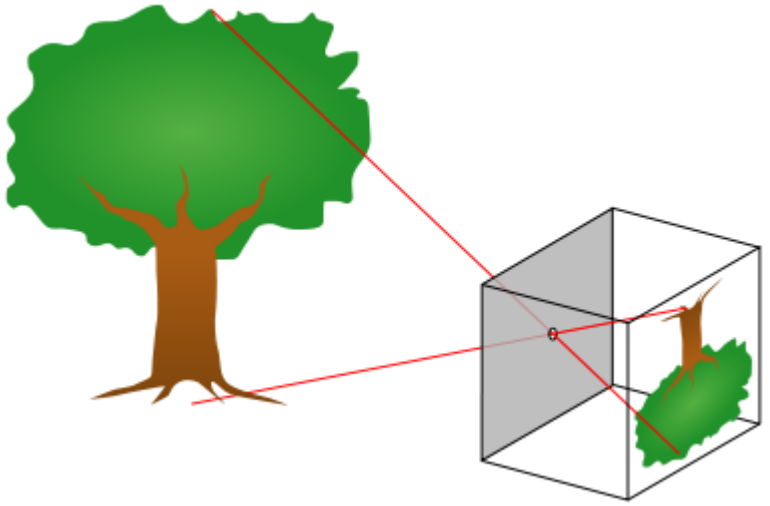
- Wiki: phot → light and graphos → drawing
  - Record light or any electromagnetic radiation
  - Chemically on a light sensitive material
  - **Electronically on an imaging sensor**
- Discretizing the usually continuous radiation (→Intensities)
- Discretizing the space domain (→Pixels)

Image Representation

- How do we capture a photo theoretically?

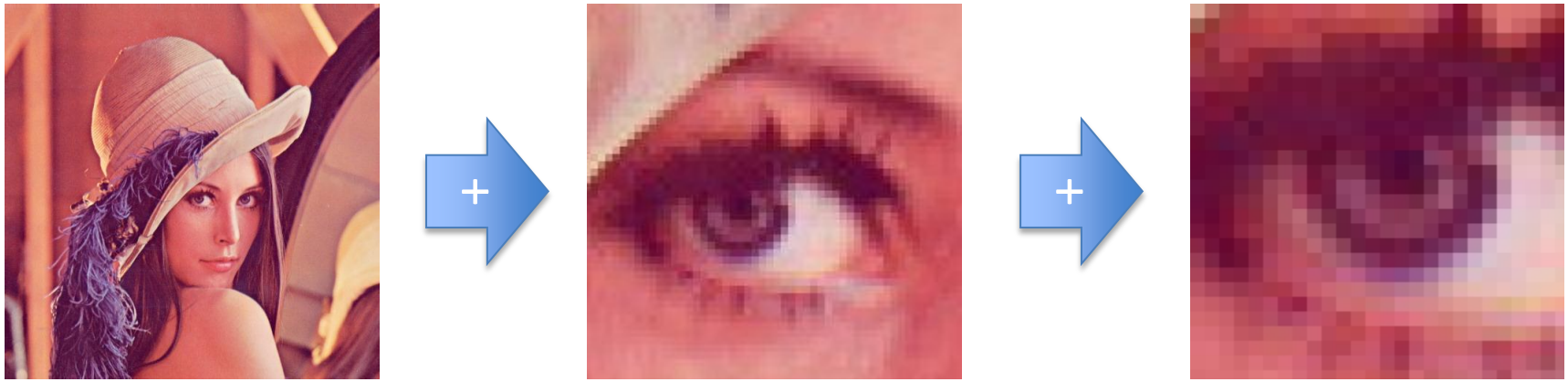
Image Representation

- How do we capture a photo theoretically?
- Pinhole camera model
  - Focal length
  - Central point
  - Distortion parameters

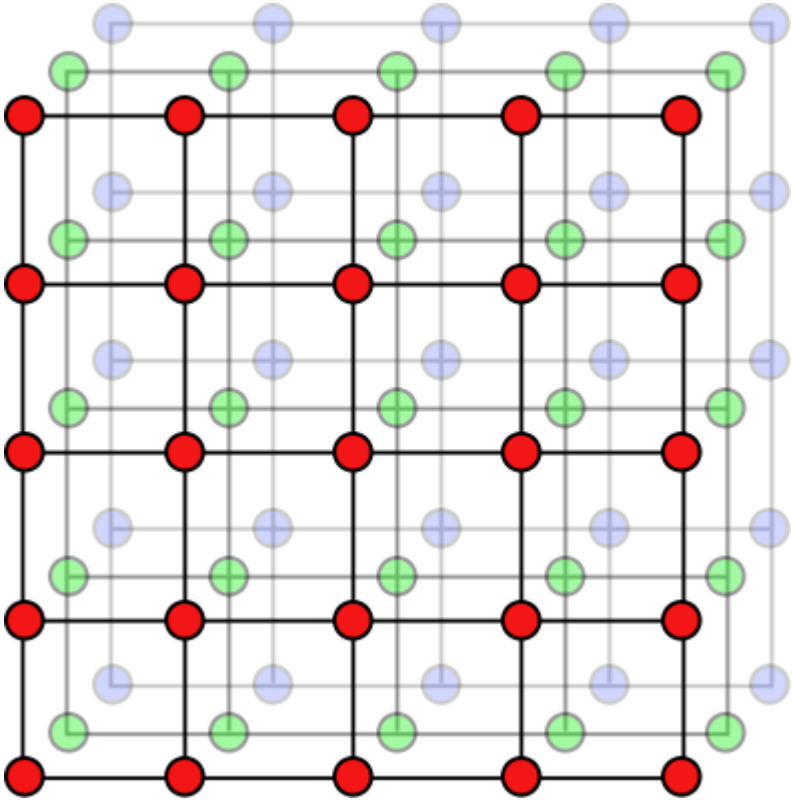


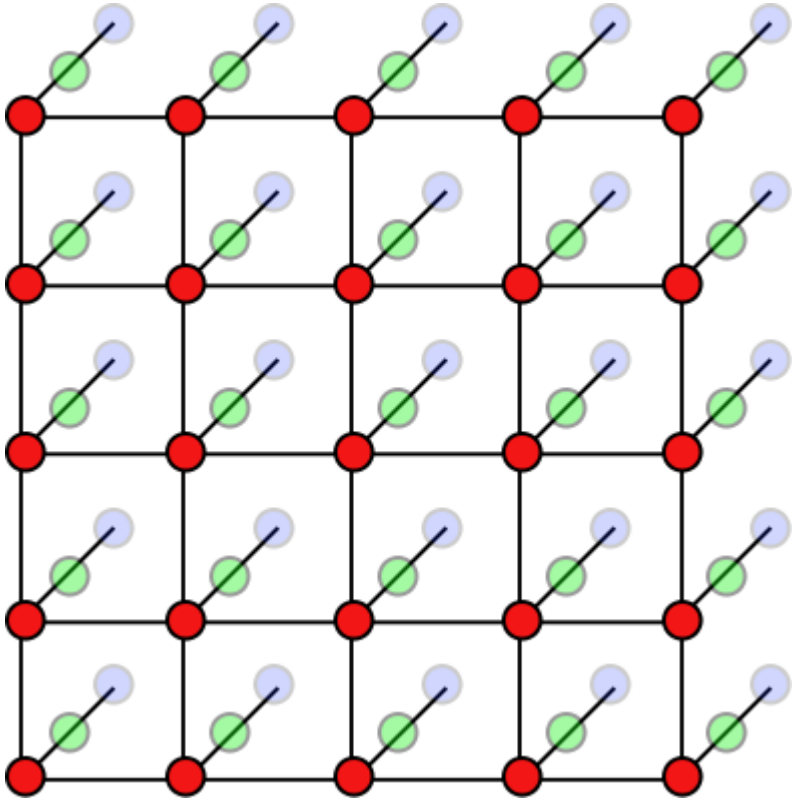
www.wikipedia.org

Image Representation



- All images can be interpreted as matrices
- Color spaces can be interpreted as
  - Different layers of matrices
  - Matrices with lists as elements





Portable Any Map (PNM)

- Includes PBM, PGM an PPM
  - Portable Bit Map
  - Portable Gray Map
  - Portable Pixel Map
- Example PBM

```
P1
# This is an example bitmap of the letter "J“
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

Portable Any Map (PNM)

- Includes PBM, PGM an PPM
  - Portable Bit Map
  - Portable Gray Map
  - Portable Pixel Map
- Example PBM

```
P1
# This is an example bitmap of the letter "J“
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

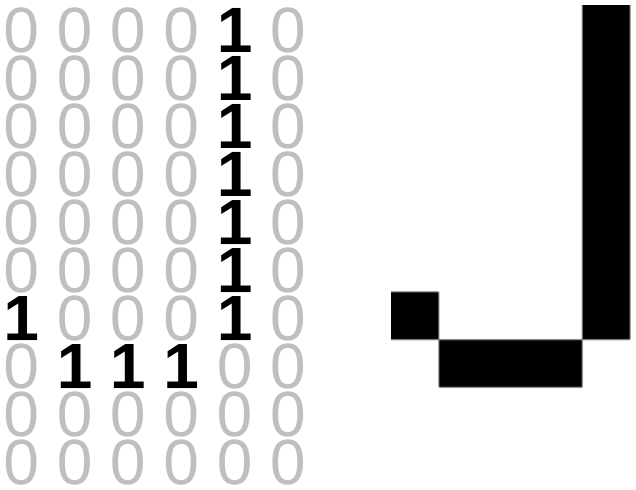
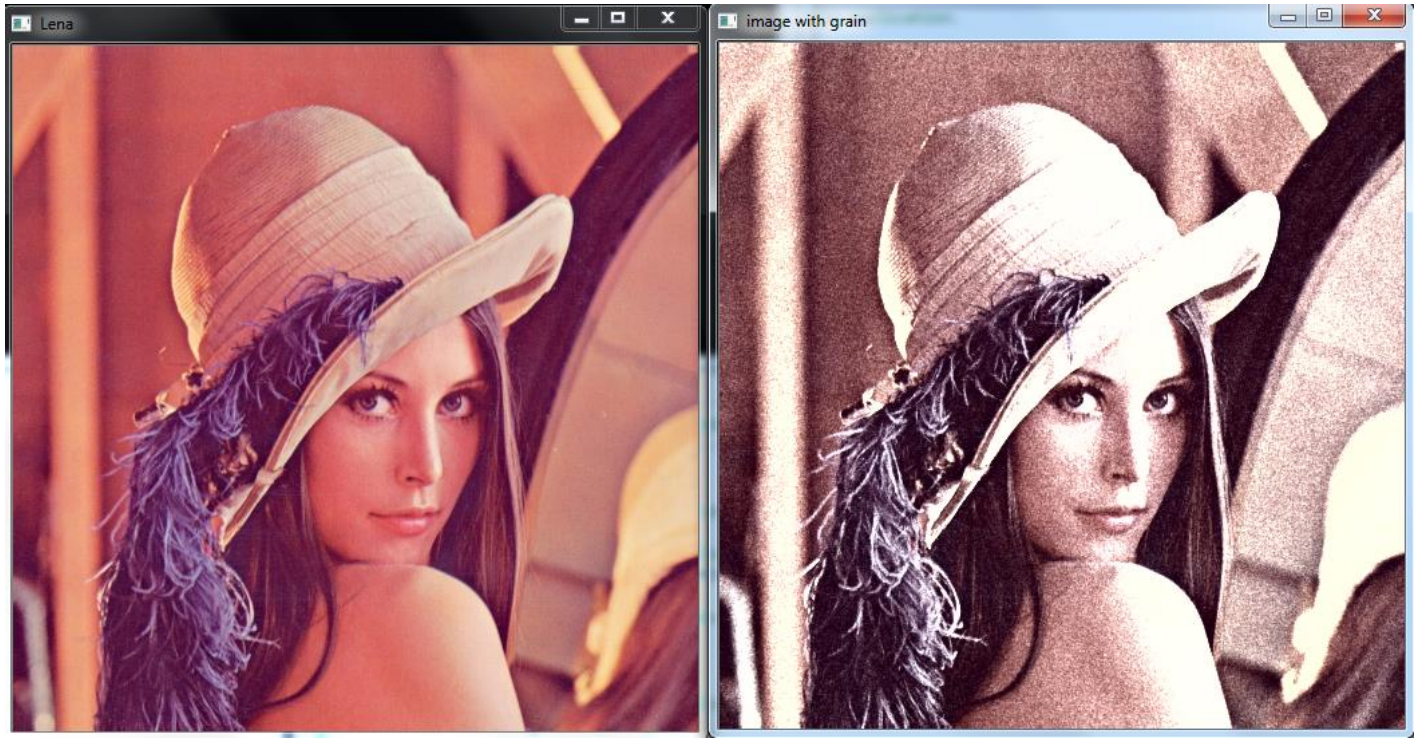


Image Manipulation

- Pixelwise manipulation in OpenCV
  - [CV-E] homepage → „OpenCV_Sample2.zip“



```
using namespace cv;

int main(int argc, char* argv[])
{
        Mat img = cv::imread("lena.jpg");
        namedWindow("Lena", 1);
        imshow("Lena", img);

        // ------------------------------------------------------------
        // -------------------    NEW SOURCE CODE    -------------------
        // ------------------------------------------------------------

        Mat img_yuv;
        // convert image to YUV color space.
        // The output image will be allocated automatically
        cvtColor(img, img_yuv, CV_RGB2YCrCb);

        // split the image into separate color planes
        vector<Mat> planes;
        split(img_yuv, planes);

        // another Mat constructor; allocates a matrix of the specified size and type;
        Mat noise(img.size(), CV_8U);

        // fills the matrix with normally distributed random values;
        randn(noise, Scalar::all(128), Scalar::all(20));

        // blur the noise a bit, kernel size is 3x3 and both sigma's are set to 0.5
        GaussianBlur(noise, noise, Size(3, 3), 0.5, 0.5);

        const double brightness_gain = 0;
        const double contrast_gain = 1.7;

        addWeighted(planes[0], contrast_gain, noise, 1, -128 + brightness_gain, planes[0]);

        const double color_scale = 0.5;
        // Scale and add values to plane[1];
        planes[1].convertTo(planes[1], planes[1].type(), color_scale, 128*(1-color_scale));

        // alternative form of convertTo if we know the datatype
        // at compile time ("uchar" here).
        // This expression will not create any temporary arrays
        // and should be almost as fast as the above variant
        planes[2] = Mat_<uchar>(planes[2]*color_scale + 128*(1-color_scale));

        planes[0] = planes[0].mul(planes[0], 1./255);

        // now merge the results back
        merge(planes, img_yuv);
        // and produce the output RGB image
        cvtColor(img_yuv, img, CV_YCrCb2RGB);
```

Thank you

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT