Visuelle Perzeption für Mensch-Maschine Schnittstellen

Vorlesung, WS 2013 / 2014

Prof. Dr. Rainer Stiefelhagen
Manuel Martinez

Institut für Anthropomatik
Karlsruhe Institut für Technologie - KIT

http://cvhci.anthropomatik.kit.edu
rainer.stiefelhagen@kit.edu
manuel.martinez@kit.edu

CVHCI Programming Assignments

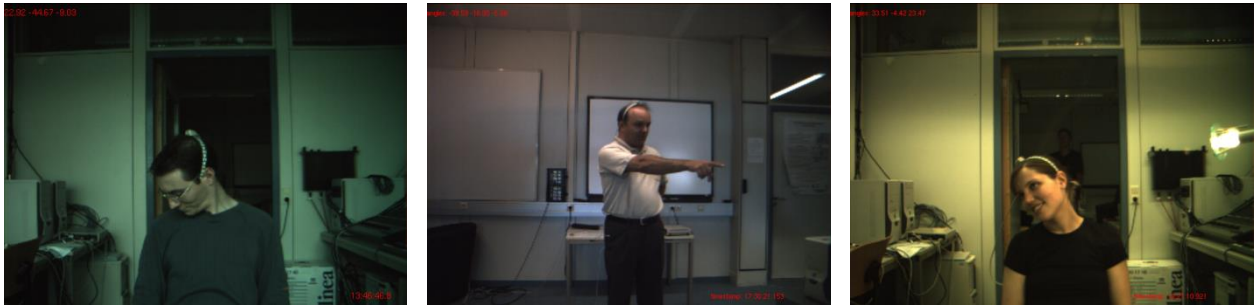
- 11.11.2013:
  - Projects: Introduction
- 2.12.2013:
  - Questions session
- 20.12.2013:
  - Questions session
- 7.2.2014:
  - Showdown and conclusions

This Lecture

- Overview of programming assignments
  - Skin Color Classification
  - Person Detection
  - Face ID
- Introduction to the suggested programming environment
- Short intro into CV programming
  - C++
    - free book: Thinking in C++ http://www.mindviewinc.com/Books/
  - OpenCV
    - Computer Vision Library
    - Documentation: http://opencv.willowgarage.com/wiki/

Skin Color Classification

- Skin-Color Detection
  - Detect skin color pixels as accurate as possible
  - Data set contains images from different lighting conditions



Data

Ground-Truth

Skin Color Classification

- Goal:
  - Develop the algorithm
  - Do a thorough quantitative evaluation
  - Visualize the results
  - Present your results in front of the class

- Do your best!
  - Let’s see the best possible results
  - Apply all tricks you can imagine

Classifying a hypothesis

- When comparing recognition hypotheses with ground-truth annotations have to consider four cases:

	Predicted positive	Predicted negative
Positive examples (Pos)	*True positive* (TP)	*False negative* (FN)
Negative example (Neg)	*False positive* (FP)	*True negatives* (TN)

- Example: Is there a cup?




Prediction: Yes No Yes No

Case: TP FN FP TN

Skin Color Classification

TP

FP

TN

FN

ROC: Receiver Operating Characteristic

- Used to evaluate binary classificators

- Measures the trade-off between true positive rate and false positive rate:

$$\text{true positive rate} = \frac{\text{TP}}{\text{Pos}} = \frac{\text{TP}}{\text{TP+FN}}$$

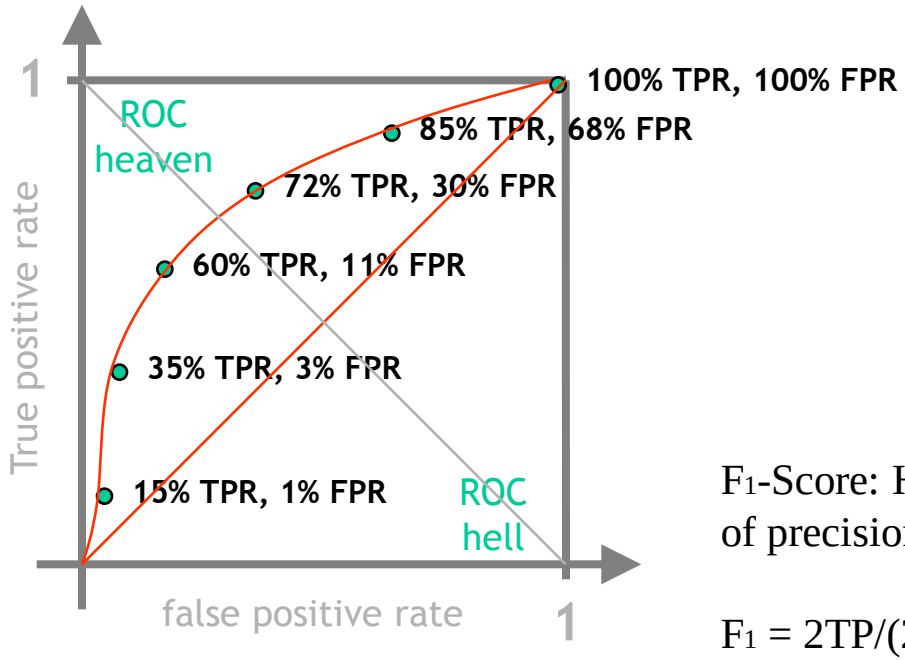
$$\text{false positive rate} = \frac{\text{FP}}{\text{Neg}} = \frac{\text{FP}}{\text{FP+TN}}$$

- Example:
  - Algorithm X classifies 80% of all skin-colored pixels correctly (true positive rate), while making 25% error on non-skin-colored pixels

ROC: Receiver Operating Characteristic

- Each prediction hypothesis has generally an associated probability value or score

- The performance values can therefore plotted into a graph for each possible score as a threshold



F_1-Score: Harmonic mean of precision and recall.

$F_1 = 2TP/(2TP+FP+FN)$

Online Judge

- http://141.3.14.100/cvhci/lecture1314/

  ← yes, it is hidden…

- It offers online evaluation of your classifier against a secret test set

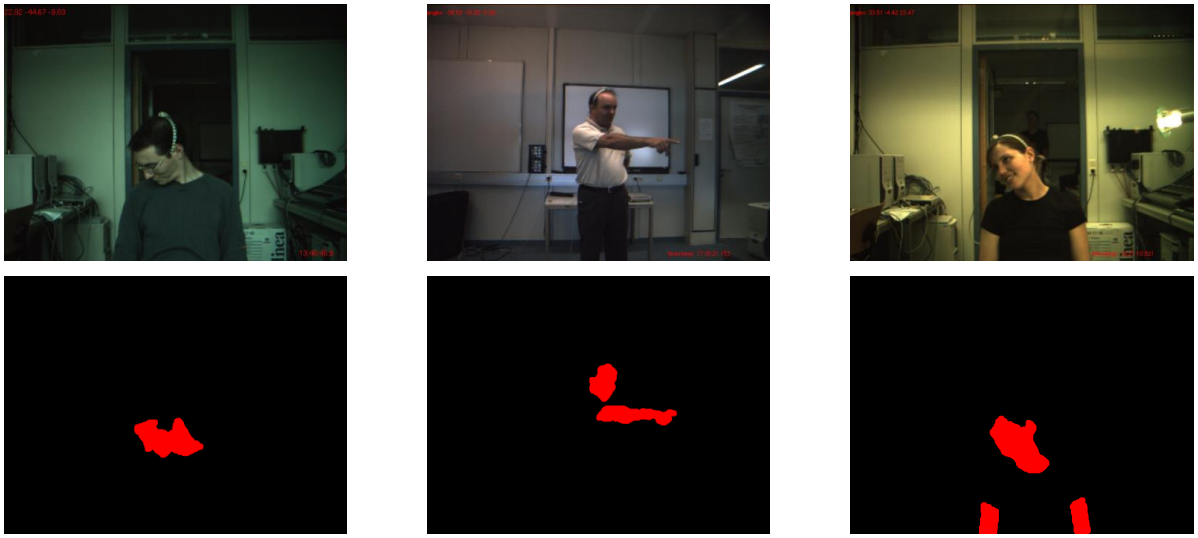
- Code + training samples are available for download on the Judge page

Assignment Files

- Folders:
  - Test (empty)
  - Train (few image samples + mask)

- assignment1.cpp
- ROC.h
- skinmodel.cpp
- skinmodel.h
- Makefile

Included data



`train/img-*.png`

`train/mask-img-*.png`

skinmodel.h

```
class SkinModel {
public:
    SkinModel();

    ~SkinModel();

    void startTraining();

    void train( const cv::Mat3b& img,
                const cv::Mat1b& mask);

    void finishTraining();

    cv::Mat1b classify(const cv::Mat3b& img);
};
```

Assignment 2: People Detection





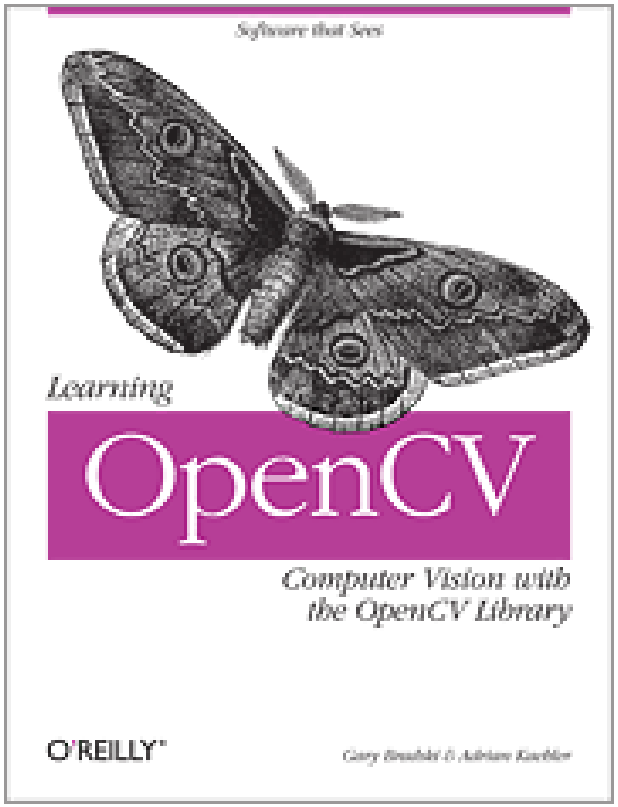
- Pre-segmented images
- Fixed scale

Assignment 3: Face ID

- 250x250 pixels
- automatically aligned
- wild faces

OpenCV

- OpenCV is an open source computer vision library containing a large number of existing algorithms
  - Image Processing:
    - Edge Detection
    - Interest Point Detection
    - Morphological Operations
  - Machine Learning
    - K-Means
    - Boosting
  - Optical Flow
  - Stereo Computation
  - Tracking
  - …



Images as matrices

```
class cv::Mat {
public:
   Mat();
  // constructs matrix of the specified size and type
  // (_type is CV_8UC1, CV_64FC3, CV_32SC(12) etc.) Mat(int
   _rows, int _cols, int _type);

   . . .
   type& at<type>(int row, int col);
  . . .
  int cols;
  int rows;
  uchar *data;
};

typedef Mat_<uchar> cv::Mat1b;
typedef Mat_<cv::vec3b> cv::Mat1b;
```

Images as matrices

- Load image (CV_8UC3)

```
cv::Mat3b img = cv::imread(fname, CV_LOAD_IMAGE_COLOR);
```

- Load image as grayscale image (CV_8UC1)

```
cv::Mat1b mask = cv::imread(fname, CV_LOAD_IMAGE_GRAYSCALE);
```

- Access image elements

```
cv::Vec3b& bgr = img(y, x);
uint8_t red    = bgr[2];
uint8_t green  = bgr[1];
uint8_t blue   = bgr[0];

uint8_t gray = mask(y, x);
```

Presentation

- **Groups of 1-3 people**
- **Date: 7.2.2014**
- Shortly present what exactly you have implemented
- Show the performance plot for different implementations / parameter choices
  - What worked best?
  - What did not work?
- What problems did you encounter?
- What were the lessons learned?
- Each group has approximately 8 minutes

End of Lecture