Visuelle Perzeption für Mensch-Maschine Schnittstellen

Vorlesung, WS 2008

Dr. Rainer Stiefelhagen
Dr. Edgar Seemann

Interactive Systems Laboratories
Universität Karlsruhe (TH)

http://isl.ira.uka.de/msmmi/teaching/visionhci
stiefel@ira.uka.de
seemann@pedestrian-detection.com

Programming

Assignments

WS 2008/09

Dr. Edgar Seemann

seemann@pedestrian-detection.com

Termine (1)

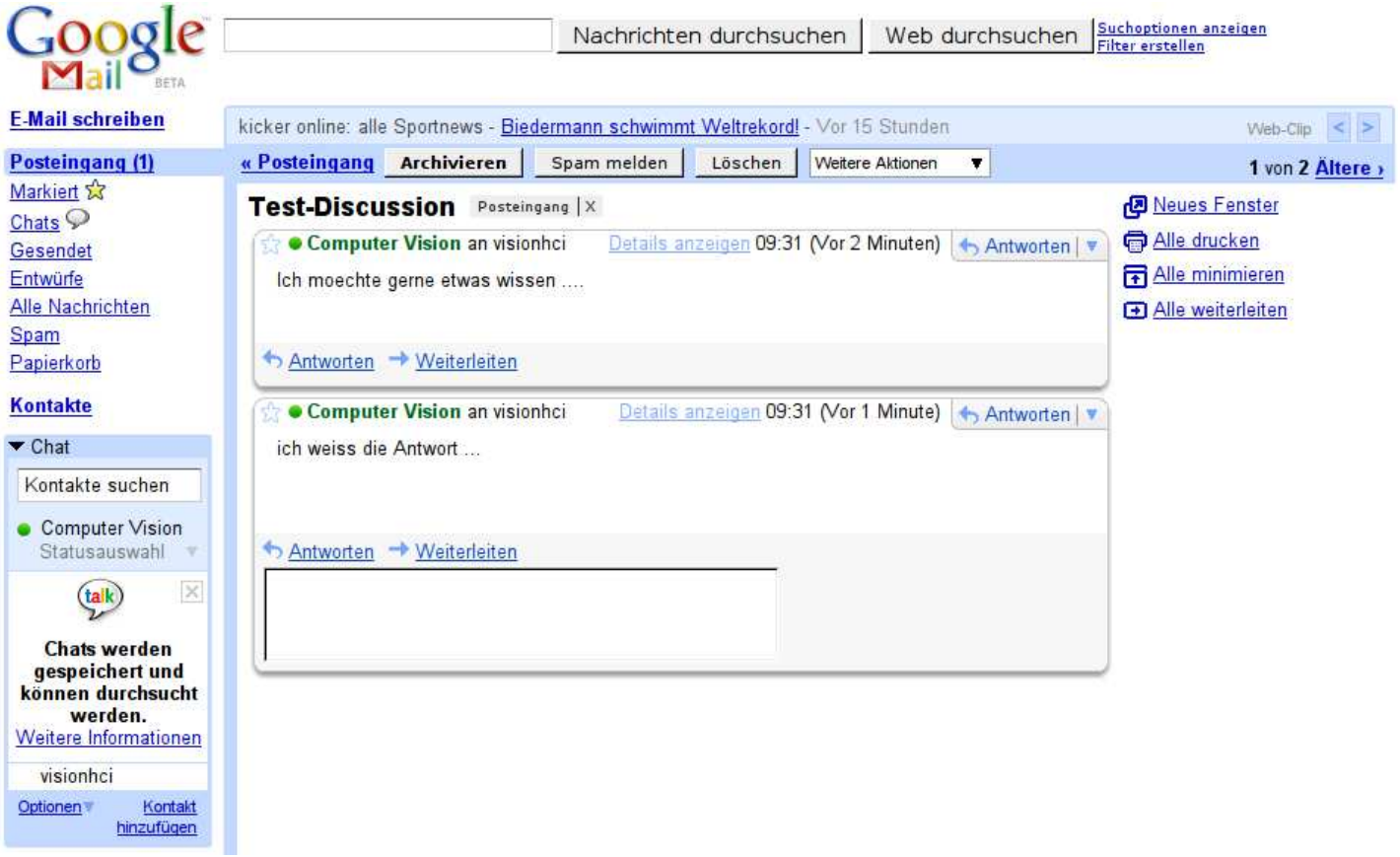
Termine	Thema
27.10.2008	Introduction, Applications
31.10.2008	Basics: Image Processing
03.11.2008	Basics: Image Transformations, 2D Structure
07.11.2008	Basics: Pattern recognition
10.11.2008	Computer Vision: Tasks, Challenges, Learning, Performance measures
14.11.2008	Face Detection I: Color, Edges (Birchfield)
17.11.2008	**Project 1: Intro + Programming tips**
21.11.2008	Project 1: Questions
24.11.2008	Face Detection II: ANNs, SVM, Viola & Jones
28.11.2008	Face Recognition I: Traditional Approaches, Eigenfaces, Fisherfaces, EBGM
01.12.2008	Face Recognition II
05.12.2008	Head Pose Estimation: Model-based, NN, Texture Mapping, Focus of Attention
08.12.2008	**Project 1: Student Presentations, Project 2: Intro**
12.12.2008	People Detection I
15.12.2008	People Detection II
19.12.2008	People Detection III (Part-Based Models)
22.12.2008	Scene Context and Geometry I (Ground-Plane, Hoiem, Leibe)

Organisatorisches

- **Gruppe 1:**
  Christian Johner
  Mike Morante
  Patrick Mehl
- **Gruppe 2:**
  Thomas Stephan (Java)
  Steffen Braun (Java)
- **Gruppe 3:**
  Martin Wagner
  Hilke Kieritz
  Jan Hendrik Hammer
- **Gruppe 4:**
  Wenlei Wu
  Chengchao Qu
- **Gruppe 5:**
  Michael Weber
  Tomas Semela
  Dennis Kopcan
- **Gruppe 6:**
  Johann Korndoerfer
  Daniel Koester
  Daniel Putsch
- **Gruppe 7**
  Benjamin Bartosch
  Thomas Lichtenstein
- **Gruppe 8**
  Florian Krupicka
  Mathias Luedtke
- **Gruppe 9**
  Igor Plotkin, Felix Reuter,
  Elke Mueller

Questions, Answers, Discussions …

- visionhci@gmail.com pwd: (see blackboard)



- Write your name at the end of your message

This Lecture

- Overview of programming assignments
- Short Intro into Programming
  - C++
    - Documentation: Thinking in C++
      http://www.mindviewinc.com/Books/
  - Qt
    - Documentation: http://doc.trolltech.com
    - Includes many tutorials

Qt Documentation

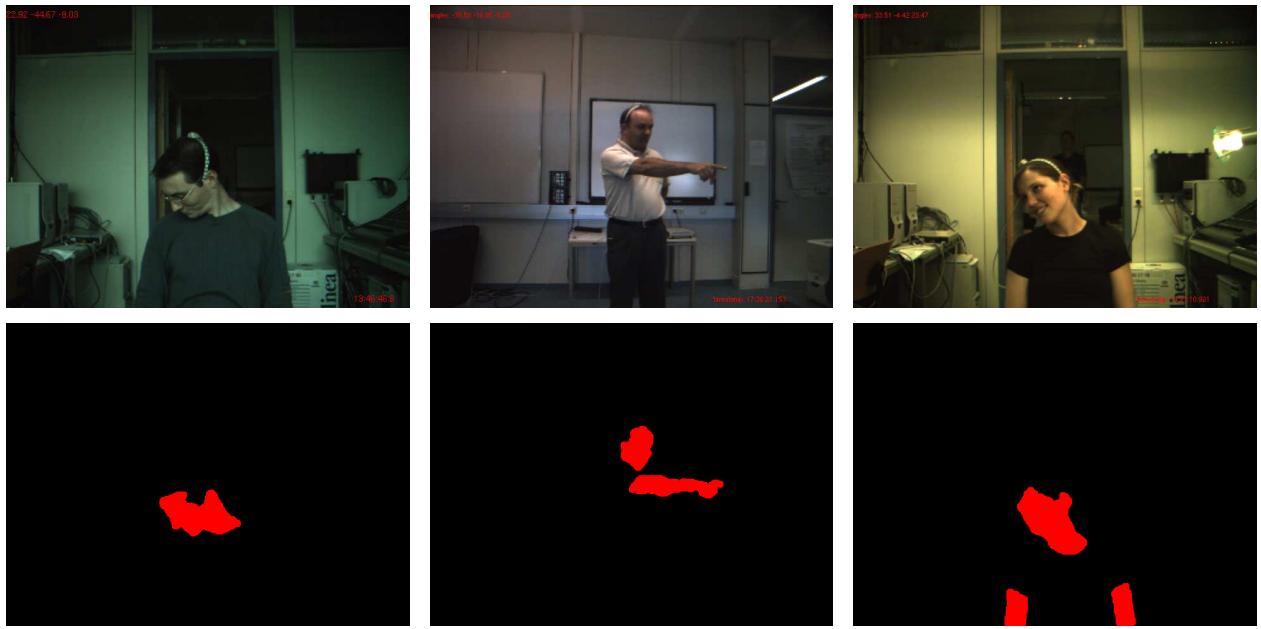
- http://doc.trolltech.com/4.4/index.html

Getting Started	General	Developer Resources
• **What's New in Qt 4.4** • How to Learn Qt • Installation • Tutorials and Examples • Porting from Qt 3 to Qt 4	• About Qt • About Us • Commercial Edition • Open Source Edition • Frequently Asked Questions	• Mailing Lists • Qt Community Web Sites • Qt Quarterly • How to Report a Bug • Other Online Resources
API Reference	**Core Features**	**Key Technologies**
• All Classes • Main Classes • Grouped Classes • Annotated Classes • Qt Classes by Module • All Namespaces • Inheritance Hierarchy • All Functions • Qt for Embedded Linux • All Overviews and HOWTOs • Qt Widget Gallery • Class Chart	• Signals and Slots • Object Model • Layout Management • Main Window Architecture • Paint System • Graphics View • Accessibility • Tool and Container Classes • Rich Text Processing • Internationalization • Plugin System • Multithreaded Programming • Inter-Process Communication (IPC) • Unit Testing Framework	• Model/View Programming • Style Sheets • Help Module • Network Module • OpenGL Module • Script Module • SQL Module • SVG Module • WebKit Integration • XML Module • XML Patterns: XQuery & XPath • Phonon Multimedia Framework • ActiveQt Framework

Assignments

Assignment 1

- Skin-Color Detection
  - Detect skin color pixels as accurate as possible
  - Data set contains 9 images from three different lighting conditions



Data

Ground-Truth

The whole thing

- Goal:
  1. Develop the algorithm
  2. Visualize the results

  3. **Do a thorough quantitative evaluation**

  4. Present your results in front of the class

It’s a competition

- View it as a competition against the other students
  - Don’t just make it work more or less
  - I want to see the best possible results
  - Apply all tricks you can imagine
  - No cheating!!!

Current directory structure

- See README.TXT

Some more details

- Training Set:
  - 3 images



  - The file trainingset.idl lists these files:
    - "/home/student/Programming/data/christian1.png";
    - "/home/student/Programming/data/cond2-alicia.png";
    - "/home/student/Programming/data/robo-edi.png";

Test Set

- Test set is defined in testset.idl (6 files)
- Ground-Truth is defined in testset-groundtruth.idl:
  - "/home/student/Programming/data/cond2-john.png": (507, 121, 508, 122):255, (508, 121, 509, 122):255, (509, 121, 510, 122):255, (510, 121, 511, 122):255 …
  - "/home/student/Programming/data/cond2-muntsin.png": (306, 202, 307, 203):255, (307, 202, 308, 203):255, (308, 202, 309, 203):255, (309, 202, 310, 203):255 …
  - "/home/student/Programming/data/petra1.png": (282, 244, 283, 245):255, (283, 244, 284, 245):255, (284, 244, 285, 245):255, (285, 244, 286, 245):255, (286, …
  - "/home/student/Programming/data/rainer1.png": (324, 180, 325, 181):255, (325, 180, 326, 181):255, (326, 180, 327, 181):255, (327, 180, 328, 181):255, (328, …
  - "/home/student/Programming/data/robo-klaus.png": (365, 95, 366, 96):255, (366, 95, 367, 96):255, (367, 95, 368, 96):255, (368, 95, 369, 96):255, (369, 95, 370 …
  - "/home/student/Programming/data/robo-rainer.png": (163, 65, 164, 66):255, (164, 65, 165, 66):255, (165, 65, 166, 66):255, (166, 65, 167, 66):255, (167, 65, 16 …

Your Task

- Produce an .idl file, which specifies for each pixel in the image, the probability of being skin colored
  - i.e. specify a 1x1 rectangle for each pixel
- Annotool helps to display results at different confidence levels

Quantitative Evaluation

- For the evaluation, we have two Python scripts
  - Directory: evaluation
    - ./fpr-rec-skin.py testset-groundtruth.idl result.idl
    - Computes true positive and false positive rate for all thresholds and writes it to plotdata.txt
  - Directory: plotting
    - ./plotSimple.py ../evaluation/plotdata.txt
    - Plots the results from plotdata.txt

Presentation

- 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

Assignment 2

- People Classification

Assignment 3

- People Detection (requires Assignment 2)

Programming Intro

C++

- Hello World

main.cpp:

```
#include <iostream> // contains cout, cin, cerr …
int main(int argc, char** argv)
{
        std::cout << “Hello World\n”;
        std::cout << “Number of arguments: “ << argc << std::endl;
        if (argc>1)
          std::cout << “First argument: “ << argv[1] << std::endl;
        return 0;
}
```

Headers and Source

- Header: defines class structure / api
- Source: the implementation

mainwindow.h:

```
#include <iostream>
class MainWindow
{
  private:
    int memberVariable1;
    int memberVariable2;
  public:
    MainWindow();//constructor
    ~MainWindow();//destructor

    void setValue1(int val);
    int getValue1();
};
```

mainwindow.cpp:

```
#include “mainwindow.h”
MainWindow::MainWindow()
{}
MainWindow::~MainWindow()
{}
void MainWindow::setValue1(int val)
{
  memberVariable1 = val;
}
int MainWindow::getValue1()
{
  return memberVariable1;
}
```

Compiling and linking

- The manual way
  - `g++ -c main.cpp mainwindow.cpp`
    - Compiles main.cpp and mainwindow.cpp into .o files
  - `g++ -o MainProgram *.o`
    - Links .o files into an executable

Qt: Creating a GUI

mainwindow.h:

```
#include <iostream>
#include <QWidget>
#include <QLabel>
#include <QVBoxLayout>

class MainWindow : public QWidget
{
  Q_OBJECT

  private:
    QLabel* imageWidget;
  public:
    MainWindow(QWidget* parent =0);

    void open(const char* file);
};
```

mainwindow.cpp:

```
#include “mainwindow.h”
MainWindow::MainWindow(QWidget*
        parent) : QWidget(parent)
{
  QVBoxLayout* layout = new QVBoxLayout();
  imageWidget = new QLabel();
  layout->addWidget(imageWidget);
  setLayout(layout);
  resize(320, 240);
  show();
}
void MainWindow::open(const char* file)
{
  QImage image(file);
  imageWidget->setPixmap(
        QPixmap::fromImage(image));
}
```

Adding window to main

main.cpp:

```
#include <iostream>
#include “MainWindow.h”
#include <QApplication>
int main(int argc, char** argv)
{
        QApplication app(argc, argv);
        MainWindow window;
        if (argc>1)
          window.open(argv[1]);
        return app.exec();
}
```

Linking libraries

- In order to use Qt, we have to link against the qt libraries
- Manual way:
  - `g++ -c *.cpp –I/path/to/headerfiles`
  - `g++ -o MainProgram *.o –L/path/to/library –lName`
- Qmake (the Qt build system):
  1. `qmake –project (create project file: dirname.pro)`
  2. `qmake (create a Makefile)`
  3. `make (execute Makefile)`
  4. `make clean (to delete built files)`
  - To add a new file/class you have to edit dirname.pro and repeat step 2 and 3

LD_LIBRARY_PATH

- -L parameter tells linker, where to look for libraries
  - `g++ -o MainProgram *.o –L/path/to/library –lName`
- Run-Time
  - Dynamically linked libraries are linked at start up
  - Dynamic libraries may be moved after linking
    i.e. we have to define search path
    - `export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/path/to/library`
  - You may define `LD_LIBRARY_PATH` in your .bashrc, then you don’t have to set it after each login
  - Search path for system libraries is typically already defined in `/etc/ld.so.conf`

Include Guards

- Avoid to include a header file multiple times

mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <iostream>
#include <QWidget>
#include <QLabel>
#include <QVBoxLayout>

class MainWindow : public QWidget
{
  Q_OBJECT

  private:
    QLabel* imageWidget;
  public:
    MainWindow(QWidget* parent =0);

    void open(const char* file);
};


#endif
```

Pointers and References

- Essentially the same thing, but
  - References cannot be null
  - References are syntactically handled as objects

- Example:
  - `QImage& img1 = open1(file);`
  - `QImage* img2 = open2(file);`
  - `img1.getPixel(0,0);`
  - `img2->getPixel(0,0);`
  - `(*img2).getPixel(0,0);`

- **Please: Avoid using pointers!!!**

Memory management

- If we create objects with new, we have to delete them
- Otherwise we have a memory leak
  - There are nice tools to detect memory leaks e.g. valgrind

- Example
  - `Object* obj = new Object()`
  - `delete obj;`
  - `Object* array = new QObject[20]; //points to the first element`
  - `delete[] array;`
- Delete is typically called in the destructor
- Exception:
  - Qt GUI elements typically use pointers, however you don’t have to worry about memory management

Tips

- Avoid calling “new”
  - `Object obj(params)`
    - Creates object in the current scope
    - Object is automatically destroyed if obj is out of scope
  - `Object* obj = new Object(params)`
    - Creates object on the heap
    - Object needs to be explicitly deleted: `delete obj;`
- Avoid objects as return values, instead pass references
  - `QImage open(const string& file)` vs.
  - `void open(const string& file, QImage& open)`
  - Removes copying overhead (even though compiler may optimize this)
  - This is also the solution to multiple return values

Const correctness

- Consider the following function signatures
  1. `open(string filename)`
  2. `open(string& filename)`
  3. `open(const string& filename)`

1. Bad: filename is passed by value, i.e. involves a copy of the string object
2. Good: filename is passed as a reference
   Bad: filename may be altered in the function
3. Assures that filename is only read in the function

Use const where ever possible

The QImage class

- QImage provides:
  - Reading and writing of various image formats
    - `QImage img(filename)`
  - Creating an empty image
    - `QImage img(w, h, QImage::Format_ARGB32)`
  - Access to image data
    - `QRgb pixel = img.getPixel(x,y)`
    - `int width = img.width()`
    - `int height = img.height()`
    - `QImage smallImage = img.scaled(w, h, Qt::AspectRatioMode, Qt::TransformationMode )`
    - `uchar* bits = img.bits()`
    - `QRgb pixel = img.getPixel(x,y)`

QRgb

- QRgb represents a RGB value
  - `QRgb pixel = qRgb(100,200,150)`
  - `int red = qRed(pixel)`
  - `int green = qGreen(pixel)`
  - `int blue = qBlue(pixel)`
- Grayscale images are stored as RGB, with r=g=b for all pixels
  - `QRgb grayPixel = qRgb(100,100,100)`

STL

- The C++ Standard Template Library provides many useful functions/classes/containers etc.
- Documentation can be found at http://www.sgi.com/tech/stl
- Examples:
  - std::vector
  - std::sort
  - std::search
- Tip: “using namespace std;” avoids the additional typing of std:: (only do this in .cpp files)

Containers

- Vectors
  - Provide dynamic arrays
  - Example:
    - `std::vector<int> numbers; //create vector of integers`
    - `numbers.push_back(5); //add 5 to vector`
    - `int val = numbers.back();`
    - `int val = numbers[0]; // array style access`
- Iterators are a generalization of pointers
  - Example:
    - `std::vector<int>::iterator it;`
    - ```
      for (it=numbers.begin(); it!=numbers.end; ++it)
        std::cout << *it;
      ```
      ```

Sorting

- Sorting
  - ```
    std::vector<double> numbers;
    …
    std::sort(numbers.begin(), numbers.end());
    ```
- Comparators/Functors
  - ```
    class compMag : public binary_function<double, double,
    bool>
    {
      bool operator()(double x, double y)
      {
        return fabs(x) < fabs(y);
      }
    };
    std::sort(numbers.begin(), numbers.end(),
    compMag());
    ```

Signals and Slots

- GUI events in Qt are handled via so-called *signals* and *slots*
- *Signals* correspond to events
- *Slots* correspond to event handlers

- signals and slots are connected by the following command:
  - `QObject::connect(button, SIGNAL(clicked()), this, clickHandler());`
- Internally Qt does some magic to make this work (you should not bother)

Example

mainwindow.cpp:

```
#include “mainwindow.h”
MainWindow::MainWindow(QWidget*
        parent = 0) : QWidget(parent)
{
  QVBoxLayout* layout = new QVBoxLayout();
  QPushButton* open = new QPushButton();
  layout->addWidget(open);
  QOjbect::connect(open, SIGNAL(clicked()), this, open());
  setLayout(layout);
  resize(320, 240);
  show();
}
void MainWindow::open()
{
  QString filename = QFileDialog::getOpenFileName(this, “Open", QDir::currentPath());
  QImage image(filename);
  imageWidget->setPixmap(QPixmap::fromImage(image))
}
```

Scripting

- Typically every computer vision tasks involves a large number of parameters, which have to be tested
- It is often extremely useful to use your programs solely from the command line with a GUI
  - Allows batch processing
  - Allows scripting
  - …
- Consequently try to separate GUI and functionality as good as possible
- Automate learning, testing
- Doing things manually does not pay off on the long run

Visualization

- Visualization often helps to understand what your code is doing (and what it is doing incorrectly)
- Possibilities:
  - Write a GUI
  - Render an image and store it to disk
  - Write data to a file and use some other tool to visualize them

libAnnotation

- Classes:
  - `AnnoRect`: represents a single annotation rectangle
  - `Annotation`: represents all annotation rectangles for an image
  - `AnnotationList`: represents a set of annotations (i.e. for a complete data set)
- Example:
  - `AnnotationList list(filename);`
  - `Annotation& anno = list[i];`
  - `AnnoRect r(x1,y1,w,h,score);`
  - `anno.add( r );`
  - `list.save(fileoutName);`
- There is also a python implementation (evaluation: AnnotationLib.py)

Plotting

- Different kinds of 2D plots are very common in computer vision
  - ROC, RPC
  - Histograms
  - etc.
- Possibilities
  - Write your own plotting routines
  - Use Qt-based plotting (e.g. QtiPlot)
  - GnuPlot
  - Matlab/Octave
  - Matplotlib (Python-based)

Matplotlib

- I have written a small script for you, which allows plotting of RPC curves:

```
./plotSimple.py data.txt
```

- Data.txt has to have the following format (and has to be sorted by score already):

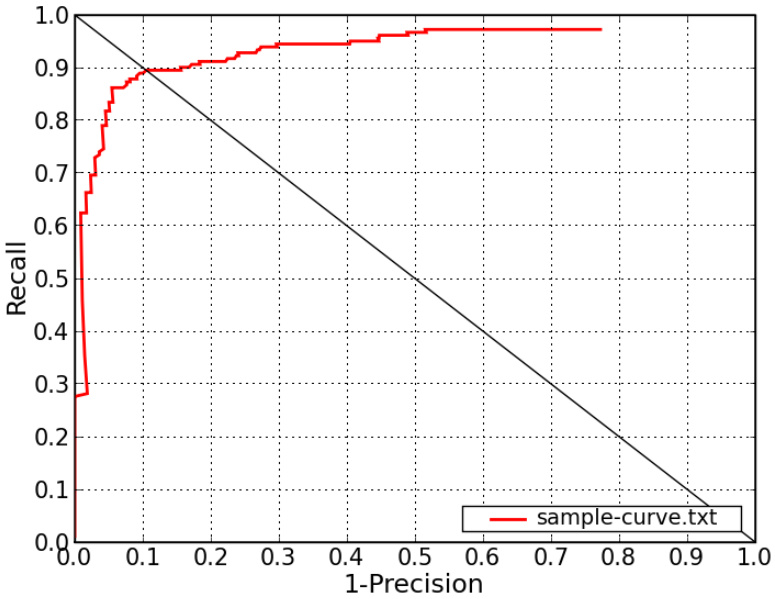
```
Column1     Column2     Column3
precision   recall      score
```

Example

```
Example:
…
…
0.919075  0.878453  245.191
0.918605  0.872928  247.271
0.923977  0.872928  247.723
0.923529  0.867403  248.576

…

…
```



End of Lecture