Visuelle Perzeption für Mensch-Maschine Schnittstellen

Vorlesung, WS 2009

Prof. Dr. Rainer Stiefelhagen
Dr. Edgar Seemann

Institut für Anthropomatik
Universität Karlsruhe (TH)

http://cvhci.ira.uka.de
rainer.stiefelhagen@kit.edu
seemann@pedestrian-detection.com
Programming

Assignments

WS 2009/10

Dr. Edgar Seemann

seemann@pedestrian-detection.com
## Termine (1)

<table>
<thead>
<tr>
<th>Termine</th>
<th>Thema</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.10.2009</td>
<td>Introduction, Applications</td>
</tr>
<tr>
<td>23.10.2009</td>
<td>Basics: Cameras, Transformations, Color</td>
</tr>
<tr>
<td>26.10.2009</td>
<td>Basics: Image Processing</td>
</tr>
<tr>
<td>30.10.2009</td>
<td>Basics: Pattern recognition</td>
</tr>
<tr>
<td>02.11.2009</td>
<td>Computer Vision: Tasks, Challenges, Learning, Performance measures</td>
</tr>
<tr>
<td>06.11.2009</td>
<td>Face Detection I: Color, Edges (Birchfield)</td>
</tr>
<tr>
<td>09.11.2009</td>
<td><strong>Project 1: Intro + Programming tips</strong></td>
</tr>
<tr>
<td>13.11.2009</td>
<td>Face Detection II: ANNs, SVM, Viola &amp; Jones</td>
</tr>
<tr>
<td>16.11.2009</td>
<td>Project 1: Questions</td>
</tr>
<tr>
<td>20.11.2009</td>
<td>Face Recognition I: Traditional Approaches, Eigenfaces, Fisherfaces, EBGM</td>
</tr>
<tr>
<td>23.12.2009</td>
<td>Face Recognition II</td>
</tr>
<tr>
<td>27.11.2009</td>
<td>Head Pose Estimation: Model-based, NN, Texture Mapping, Focus of Attention</td>
</tr>
<tr>
<td>30.11.2009</td>
<td>People Detection I</td>
</tr>
<tr>
<td>03.12.2009</td>
<td>People Detection II</td>
</tr>
<tr>
<td>07.12.2009</td>
<td><strong>Project 1: Student Presentations, Project 2: Intro</strong></td>
</tr>
<tr>
<td>11.12.2009</td>
<td>People Detection III (Part-Based Models)</td>
</tr>
<tr>
<td>14.12.2009</td>
<td>Scene Context and Geometry</td>
</tr>
<tr>
<td>18.12.2009</td>
<td>Stereo and Optical Flow</td>
</tr>
<tr>
<td>21.12.2009</td>
<td>TBA</td>
</tr>
</tbody>
</table>

Edgar Seemann, 09.11.2009
Performance Measures
Performance Measures

- Measuring the performance of object recognition algorithms is not trivial
- There are different measures depending on the application

1. For classification (i.e. yes/no decision, if object is present or not)
   - ROC (Receiver-Operating-Characteristic)

2. For localization (i.e. detecting the object’s position)
   - RPC (Recall-Precision-Curve)
   - DET (Detection Error Trade-Off)
Classifying a hypothesis

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

<table>
<thead>
<tr>
<th>Predicted positive</th>
<th>Predicted negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive examples (Pos)</td>
<td>True positive (TP)</td>
</tr>
<tr>
<td>Negative example (Neg)</td>
<td>False positive (FP)</td>
</tr>
</tbody>
</table>

- Example:
ROC

- Used for the task of classification
- Measures the trade-off between true positive rate and false positive rate:

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

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

- Example:
  - Algorithm X detects 80% of all cups (true positive rate), while making 25% error on images not containing cups
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
Recall-Precision

- For localization ROC is not appropriate, since the number of hypothesis varies
- We therefore define:
  - Recall: percentage of objects found
  - Precision: percentage of correct hypotheses

\[
\begin{align*}
\text{true positive rate} & = \frac{TP}{Pos} = \frac{TP}{TP+FN} \\
\text{false positive rate} & = \frac{FP}{Neg} = \frac{FP}{FP+TN} \\
\text{recall} & = \frac{TP}{Pos} \\
\text{precision} & = \frac{TP}{TP+FP} = \frac{TP}{\#\text{hypotheses}}
\end{align*}
\]
Plotting Recall-Precision

- RPC are typically plotted on a recall vs. 1-precision scale:

  ![Plotting Recall-Precision Diagram](image)

  - 72% TPR, 70% precision
  - 70% recall, 68% precision
  - 60% recall, 89% precision
  - 35% TPR, 97% precision
  - 15% recall, 99% precision
Detection Error Trade-Off

- DET measures the number of false detections per tested image window with respect to the miss-rate (1 – recall)
- Used for a sliding window based detector

Disadvantages:
- Chart more difficult to read (e.g. log-scale)
- Depends on the number of windows tested (i.e. image size, sliding window parameters)
- Does not measure the performance of the complete detection system including non-maximum suppression
Further measures

- In order to express the performance in a single figure, the following measures are common:
  - **Equal Error Rate (EER)**
    - The point where the errors for true positives and false positives are equal (i.e. points on the diagonal from (0,1) to (1,0))
    - Not well-defined for RPC
  - **Area Under Curve (for ROC)**
    - The area under the curve ;-) 
  - **Mean Average Precision (for RPC)**
    - Average precision values
    - Sometimes measured only at pre-defined recall values
  - **False Positives per Image (FPPI)**
Tracking Measures

- For tracking there are performance measures considering object id changes [Bernadin&Stiefelhagen 2008]

The multiple object tracking precision (MOPT):

The multiple object tracking accuracy (MOTA):
Localization and Ground-Truth

- For localization, the test data is mostly annotated with ground-truth bounding boxes
- It is often not obvious when to count a hypotheses as true or false detection
  - Misaligned hypotheses
  - Double detections
Comparing hypotheses to Ground-Truth

- **Comparison measures**
  1. Relative Distance
  2. Cover and Overlap

1. Relative distance

\[
d_r = \sqrt{\left(\frac{2 \cdot \Delta x}{w}\right)^2 + \left(\frac{2 \cdot \Delta y}{h}\right)^2}
\]
Comparing hypotheses to Ground-Truth

2. Cover and Overlap

- There is no standard for which values to choose
- Sometimes used as threshold:
  - Cover > 50%, Overlap > 50%, Relative Distance < 0.5
- Double detections are counted as false positive
Assignments
Organisatorisches

- Gruppe 3
  - Tingyun, Zhang
  - Ning, Zhu
- Gruppe 4
  - Andreas, Wachter
  - Sina, Martens
  - Sijie, Shen
- Gruppe 8
  - Alexander, Herzog
  - Stefan, Bürger
  - Jonathan, Wehrle
- Gruppe 9
  - Marc, Essinger
- Gruppe ?
- Gruppe 5
  - Christian, Wittner
  - Matthias, Steiner
  - Christoph, Weber
- Gruppe 6
  - Nils, Adermann
  - Jan, Issac
- Gruppe 7
  - Alexander, Wirth
  - Dimitri, Majarle
  - Paul, Märgner
- Gruppe 1
  - Patrick, Grube
  - Benjamin, Hohl
  - Bastiaan, Hovestreydt
- Gruppe 2
  - Sebastian, Bodenstedt
  - Michael, Heck
Questions, Answers, Discussions …

- Mailing List
  - cvhci09@ira.uni-karlsruhe.de
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


<table>
<thead>
<tr>
<th>Getting Started</th>
<th>General</th>
<th>Developer Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What's New in Qt 4.4</strong></td>
<td><strong>About Qt</strong></td>
<td><strong>Mailing Lists</strong></td>
</tr>
<tr>
<td>How to Learn Qt</td>
<td>About Us</td>
<td>Qt Community Web Sites</td>
</tr>
<tr>
<td>Installation</td>
<td>Commercial Edition</td>
<td>Qt Quarterly</td>
</tr>
<tr>
<td>Tutorials and Examples</td>
<td>Open Source Edition</td>
<td>How to Report a Bug</td>
</tr>
<tr>
<td>Porting from Qt 3 to Qt 4</td>
<td>Frequently Asked Questions</td>
<td>Other Online Resources</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>API Reference</th>
<th>Core Features</th>
<th>Key Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Classes</td>
<td>Signals and Slots</td>
<td>Model/View Programming</td>
</tr>
<tr>
<td>Main Classes</td>
<td>Object Model</td>
<td>Style Sheets</td>
</tr>
<tr>
<td>Grouped Classes</td>
<td>Layout Management</td>
<td>Help Module</td>
</tr>
<tr>
<td>Annotated Classes</td>
<td>Main Window Architecture</td>
<td>Network Module</td>
</tr>
<tr>
<td>Qt Classes by Module</td>
<td>Paint System</td>
<td>OpenGL Module</td>
</tr>
<tr>
<td>All Namespaces</td>
<td>Graphics View</td>
<td>Script Module</td>
</tr>
<tr>
<td>Inheritance Hierarchy</td>
<td>Accessibility</td>
<td>SQL Module</td>
</tr>
<tr>
<td>All Functions</td>
<td>Tool and Container Classes</td>
<td>SVG Module</td>
</tr>
<tr>
<td>Qt for Embedded Linux</td>
<td>Rich Text Processing</td>
<td>WebKit Integration</td>
</tr>
<tr>
<td>All Overviews and HOWTOs</td>
<td>Internationalization</td>
<td>XML Module</td>
</tr>
<tr>
<td>Qt Widget Gallery</td>
<td>Plugin System</td>
<td>XML Patterns: XQuery &amp; XPath</td>
</tr>
<tr>
<td>Class Chart</td>
<td>Multithreaded Programming</td>
<td>Phonon Multimedia Framework</td>
</tr>
<tr>
<td></td>
<td>Inter-Process Communication (IPC)</td>
<td>ActiveQt Framework</td>
</tr>
<tr>
<td></td>
<td>Unit Testing Framework</td>
<td></td>
</tr>
</tbody>
</table>

Edgar Seemann, 09.11.2009
Assignment 1

- Skin-Color Detection
  - Detect skin color pixels as accurate as possible
  - Data set contains images from 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:
  - 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";
    - ...

Edgar Seemann, 09.11.2009
Test Set

- Test set is defined in testset.idl
- Ground-Truth is defined in testset-groundtruth.idl:
  - "/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, ...
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
    - \texttt{./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
    - \texttt{./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
VirtualBox: getting data in/out

- Internet access should work through NAT, if not:
  - `ifconfig -a` //shows all network interfaces
  - `sudo dhclient ethX` //request IP address
  - alternatively edit `/etc/network/interfaces`

- Shared Folders
  - Menu -> Devices -> Shared Folders -> Machine Folders
  - Mount folder:
    - `sudo mount.vboxsf SHARE_NAME /mnt`
  - alternatively edit `/etc/fstab`
Assignment 2

- People Classification
Programming Intro
C++

Hello World

main.cpp:

```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 {

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);

    void open(const char* file);

};

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

#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)
OpenCV

- Okapi is based on OpenCV 2.0
- 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
    - SVM
    - Boosting
  - Optical Flow
  - Stereo Computation
  - Tracking
Images as matrices

- **Load image**
  
  ```cpp
  Mat img = imread(image_fname);
  ```

- **Load image as grayscale image**
  
  ```cpp
  Mat maskimg = imread(mask_fname, CV_LOAD_IMAGE_GRAYSCALE);
  ```

- **Access image elements**
  
  ```cpp
  int pixelValue = img.at<uchar>(y,x);
  Vec3b& pixelValue = img.at<Vec3b>(y,x);
  int red = pixelValue[2];
  int green = pixelValue[1];
  int blue = pixelValue[0];
  ```
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
  - \( \text{std::vector<double> numbers;} \)
  - \( \ldots \)
  - \( \text{std::sort(numbers.begin(), numbers.end());} \)

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

```cpp
QObject::connect(button, SIGNAL(clicked()), this, clickHandler());
```

Internally Qt does some magic to make this work (you should not bother)
Example

mainwindow.cpp:

```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))
}
```

Edgar Seemann, 09.11.2009
Scripting

- Typically every computer vision task 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**

<table>
<thead>
<tr>
<th>Recall</th>
<th>1-Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.919075</td>
<td>0.878453</td>
</tr>
<tr>
<td>0.918605</td>
<td>0.872928</td>
</tr>
<tr>
<td>0.923977</td>
<td>0.872928</td>
</tr>
<tr>
<td>0.923529</td>
<td>0.867403</td>
</tr>
</tbody>
</table>

...
End of Lecture