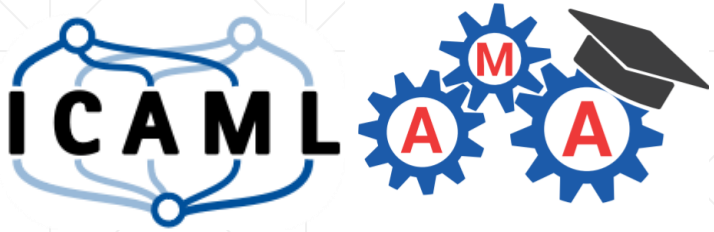
Interdisciplinary
Center for Applied
Machine Learning



Applied
Machine Learning
Academy

Programming Languages and Frameworks for Data Science

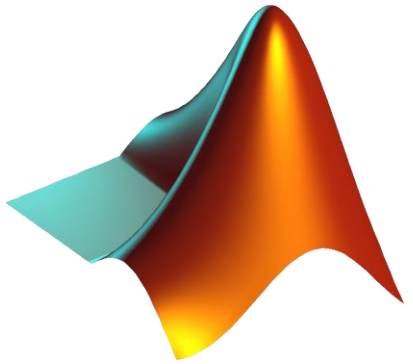
AMA / ICAML - 01.10.2019

Programming languages
Second part

MATLAB / Octave

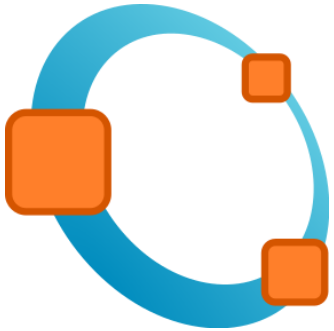
Advantages

- Matrix-centered multi-purpose programming
- Very good documentation, wide usage in the field
- Extensible
- High-quality toolboxes (however, expensive!) for MATLAB



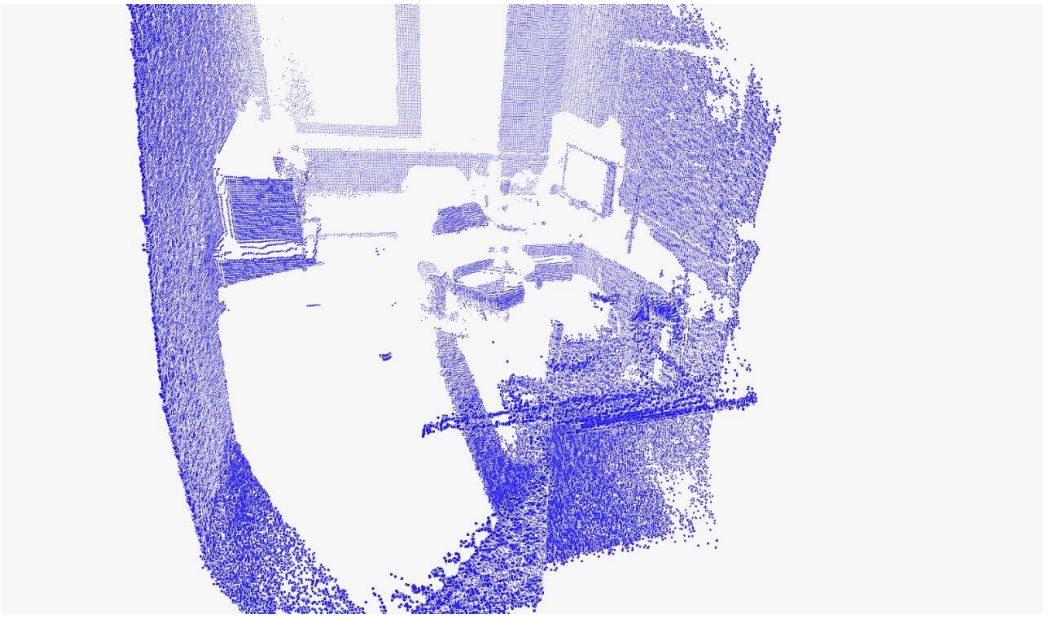
Drawbacks

- Expensive
- Non Open Source
- Open-Source version Octave is not fully equivalent



Matlab Example – 3D Points

Depth image to point cloud

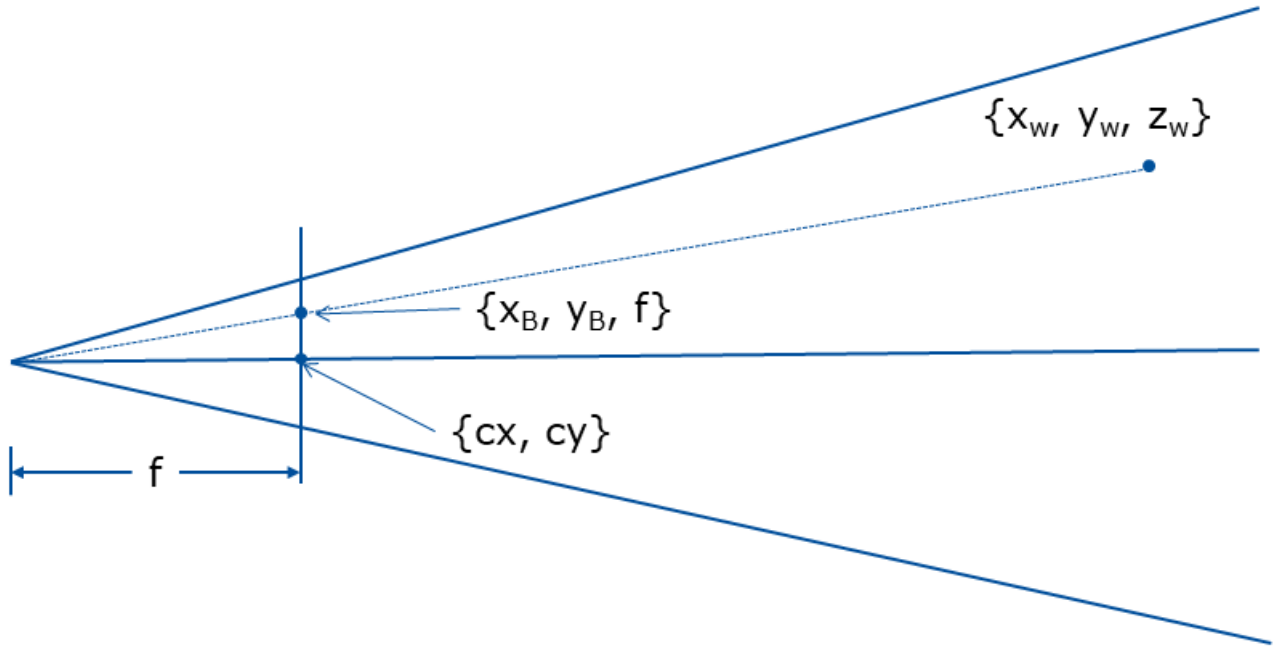


Matlab Example – 3D Points

```
function [ pointCloud, pointToImgId ] = convertToPointCloud(image, cx, cy, f)
    [y_size, x_size] = size(image);
    [x,y] = meshgrid(1:size(image,2),1:size(image,1));

    zw = (double(-(y-cy)).*image) / f;
    xw = (double((x-cx)).*image) / f;
    new_size = x_size * y_size;

    xw_1d = reshape(xw, new_size, 1);
    yw_1d = reshape(image, new_size, 1);
    zw_1d = reshape(zw, new_size, 1);
...
```



Matlab Example – 3D Points

```
...
        pointToImgId = transpose(1:1:(y_size*x_size));
       zero = find(yw_1d == 0);
       xw_1d(zero) = [];
       yw_1d(zero) = [];
       zw_1d(zero) = [];
       pointToImgId(zero) = [];

       pointCloud = [floor(xw_1d), floor(yw_1d), floor(zw_1d)];
end


[data, pointToImgId] = convertToPointCloud(img, cx, cy, f);
```

Matlab - Data Science Prozess

Data Acquisition	Exploration / Preprocessing	Modelling	Interpretation & Presentation	Production, Deployment
	• Native N-dimensional array object • Statistics & ML Toolbox	• Statistics & ML Toolbox • Deep Learning Toolbox	• Native Visualization	• <c++ gen>

Matlab Universe

- Native support for high performance N-dimensional array object calculations
  - Reach set of numerical algorithms
  - Parallelization support

- Native visualization
  - Easy to use
  - Well integrated in the DIE

- Statistics & ML Toolbox
  - Descriptive Statistics
  - Classification
  - Regression
  - Clustering

C++

Advantages

- High performance
- Extremely high-quality libraries (boost)
- Platform-independence even towards GPU and Embedded
- Embeddable into Python, Java, R and MATLAB (almost anywhere)
- full support for generic programming
- very modern standard (C++17 is ready)

Drawbacks

- Compiler errors are difficult to read (especially, when using generics)
- Some inconsistencies between compilers
- Memory management
- Initial learning cost

C++ - Data Science Prozess

Datenbe-schaffung	Exploration / Preprocessing	Modellierung	Interpretation & Präsentation	Production, Deployment
	• opencv			

The Backend

Java

Advantages

- Good performance
- High-quality Design and Runtime
- Platform-independence
- Easy to learn (very good error messages)
- Safe memory management

Drawbacks

- Unable to unlock some aspects of modern computers (GPUs, specific instructions)
- Overhead produced by memory management
- Oracle licensing strategy

Java - Data Science Prozess

Data Acquisition	Exploration / Preprocessing	Modelling	Interpretation & Presentation	Production, Deployment
	• OpenIMAJ	• **Weka** • **RapidMiner** • Mallet • Deep Learning for Java (DL4J) • Tensorflow	• JFreeChart	• Java EE

Java Universe

- Weka
  - GUI
  - Large Model/Algorithm Library
  - Opensource
  - For the scientist by the scientist

- RapidMiner
  - GUI
  - Large Model/Algorithm Library
  - Opensource
  - Commercial software

rapidminer

Python

Advantages

- Nice, modern scripting language
- Huge amount of software available
- C++ friendly (easy to extend towards high performance)
- Compact code, due to syntactical sugar

Drawbacks

- Difficult to read, due to syntactical sugar
- Large number of ML and DS libraries
- Software Quality (especially packages) varies
- Easy to break: Virtual environment stuff, versions, python2 vs. Python3
- Slow

Teil 1 - face_detect.py

```
import cv2
import sys

# Get user supplied values
imagePath = sys.argv[1]
cascPath = sys.argv[2]

# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)

# Read the image
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Teil 2 - face_detect.py

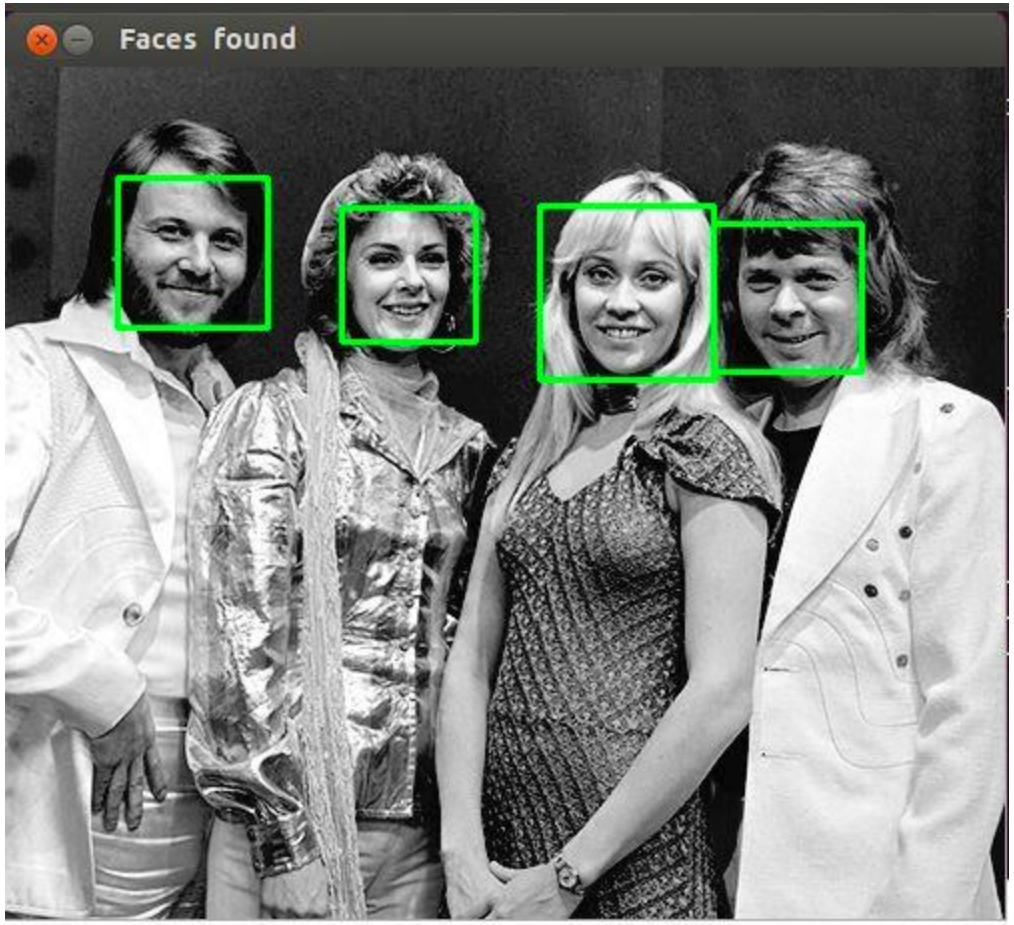
```
#Detect faces in the image.
faces = faceCascade.detectMultiScale(
gray,
scaleFactor=1.1,
minNeighbors=5,
minSize=(30, 30)
)
print(faces)
print("Found {0} faces!".format(len(faces)))

# Draw a rectangle around the faces
for (x, y, w, h) in faces:
cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.imshow("Faces found", image)
cv2.waitKey(0)
```

Face Detection in Python

The source code is

- easy to **read**
- easy to **modify**
- Complex algorithms made accessible for anyone
- Performance overhead can be ignored.



Python - Data Science Prozess

Data Acquisition	Exploration / Preprocessing	Modelling	Interpretation & Presentation	Production, De ployment
• Tweetpy • scrapy	• **Numpy** • **Pandas** • Opencv	• **Scikit** --Deep Learning-- • Kereas • TensorFlow • Pytorch • Caffee	• **Matplotlib** • Seaborn • Ipyvolume • folium	• Flusk

SciPy Univers

- Numpy:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

- Pandas
  - Data frames with reach functionality
  - Visualization
  - Join operations

SciPy Univers

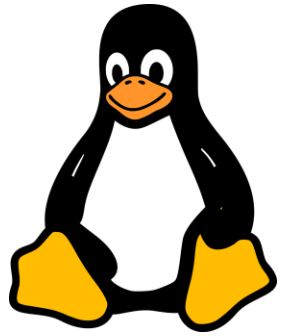
- Scikit
  - Simple and efficient tools
  - Data Mining
  - Data Analysis
  - Machine Learning

- Matplotlib
  - 2D plotting
  - Publication quality figures
  - Integration with IPython, Jupyter

Unix - Bash

Advantages

- Batch Processing
- Platform independency
- Open Source
- Containerization
- Large number of opensource tools
- Bash scripts



Drawbacks

- Missing expertise
- Terminal interaction not tracked, reviewed or versioned

Unix jq example

$ cat tweets.json:


```
{
  "contributors": null,
  "truncated": true,
  "text": "The Shortest Paths Dataset used for ...",
  "is_quote_status": false,
  "in_reply_to_status_id": null,
  "id": 1062405858712272900,
  "favorite_count": 3,
  "source": "<a href=\"http://twitter.com/download/an...",
  "retweeted": false,
  "coordinates": null,
  "entities": {
    "symbols": [],
    "user_mentions": [],
    "hashtags": [
      {
...
```


$ cat tw.csv:


```
516914567717617660,"osmfilter and o",2
511380240506306560,"RT @calestous: ",0
506544606348328960,"Need some clean",1
504522183914569700,"New page design",2
...
```


```
cat tweets.json | jq -r "[.id, (.text | .[0:15]), (.entities.hashtags | length)] | @csv" > tw.csv
```

Unix - Data Science Prozess

Data Acquisition	Exploration / Preprocessing	Modelling	Interpretation & Presentation	Production, Deployment
	• Jq • Csvkit • Image-magic			• **Docker** • Kubernetes

Unix Universe

- Docker / Containerization
  - Security
  - Scalability
  - Dependency reduction

- Terminal tools
  - fast processing
  - Simplicity

Wrap-Up

- Python: A useful scripting language with high adoption rate, but sometimes easy to break
- R: A fully functional data science environment that feels like a classical imperative scripting language
- MATLAB and Octave: If you need matrices and matrix algebra, then consider MATLAB and Octave.
- C++: You need to scale up to unlimited performance still using a high-quality, nice language: (Modern!) C++ is here for you.
- Java: You need to scale out? Java is the way to go. Not the fastest, not the most efficient, but easy to use and not so error-prone...