Algorithm development and evaluation with virtual environments

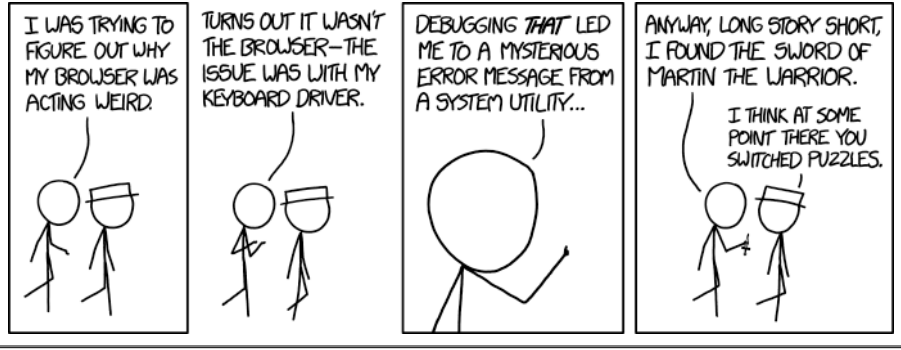
Dipl.-Inf. Robert Manthey

Juniorprofessur MEDIACOMPUTING
TECHNISCHE UNIVERSITÄT CHEMNITZ

localizeit
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
INTENTA
ADVANCED RECOGNITION COMPONENTS
3D MICROMAC
3DInsight.de
Your Visualization Partner
IBS
IBS Software & Research GmbH

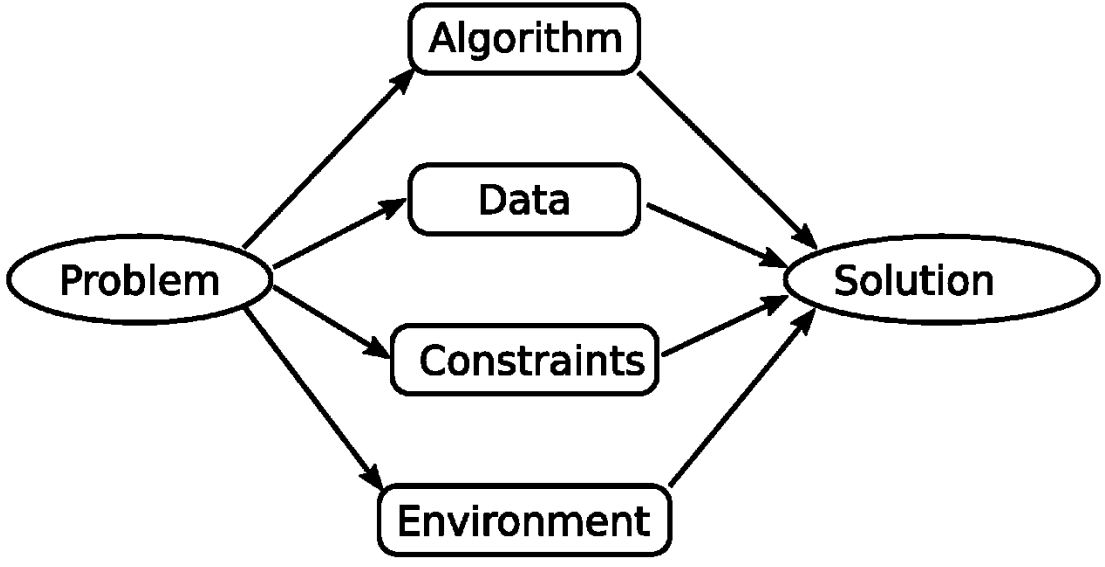
Dipl.-Inf. Robert Manthey

- 2008: Univerity degree in computer science at Technical University of Chemnitz
- Since 2008: Different software projects at database management, ergonomics, facility-and-school safety, automotive embedded system testing, media processing and image understanding
- 2017: Guest research at NIST



[1]

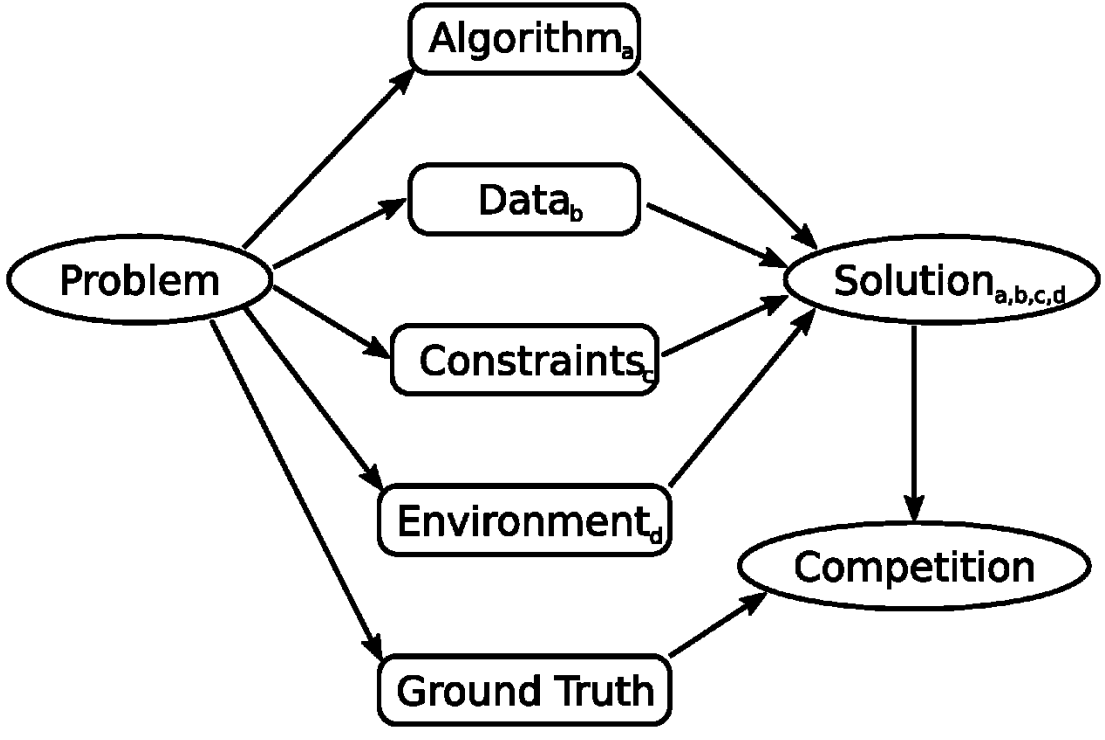
Background & Problems (I)



Solving a problem need a *Algorithm* to process the *Data* and handle the *Constraints* in given *Environment*.

BUT

Background & Problems (II)

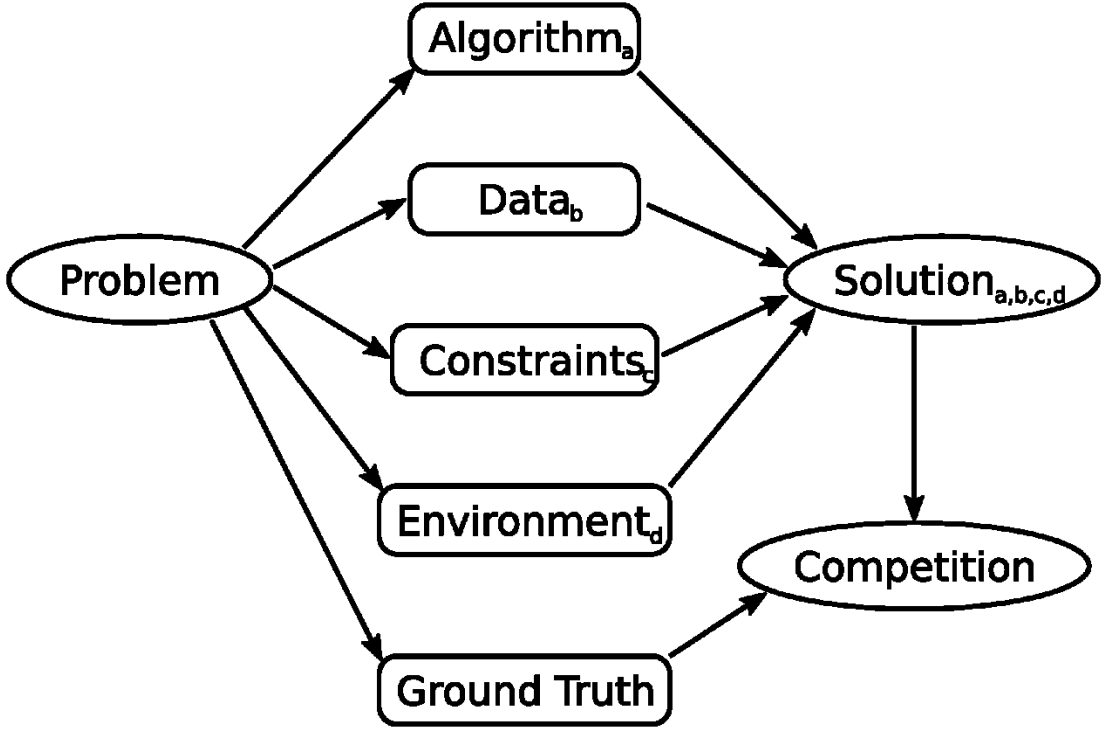


Solving a problem need a *Algorithm* to process the *Data* and handle the *Constraints* in given *Environment*.

BUT

Verifications, Evaluations and Competitions need similar elements.

Background & Problems (III)



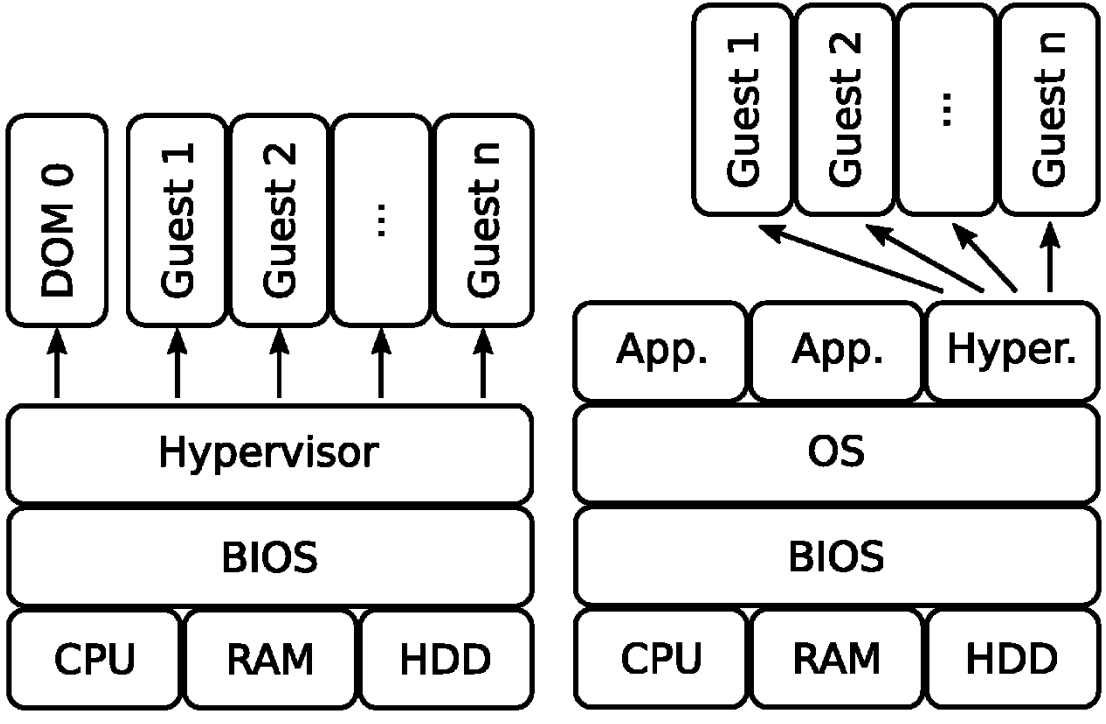
Often only *Algorithm*, *Data* and related informations being provided.

BUT, no informations of

- *Environment*-dependencies to external programs or libraries, environment variable settings, functions from older versions of source code etc.
- *Constraints* to specific libraries, certain hardware/ GPU, parameter settings etc.

→ Much “Debugging“ needed

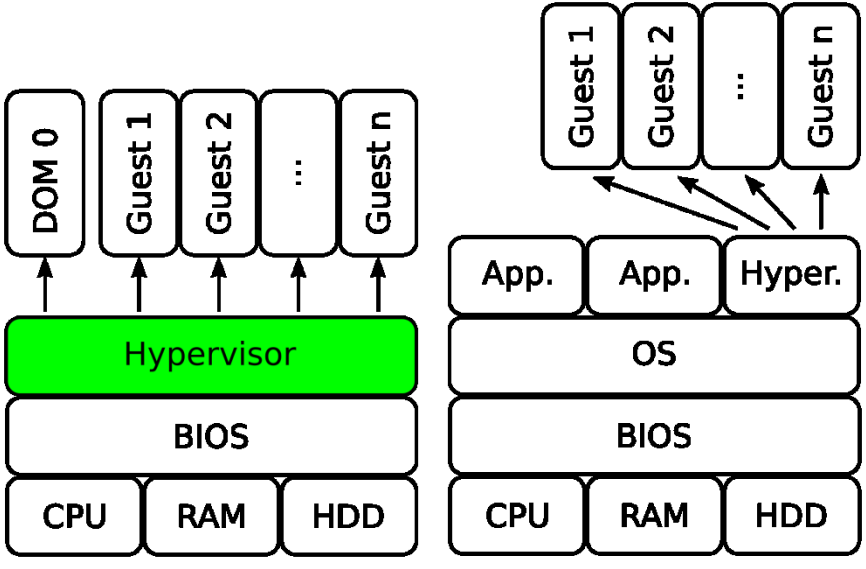
Virtual Machines



Virtual Machines provide:

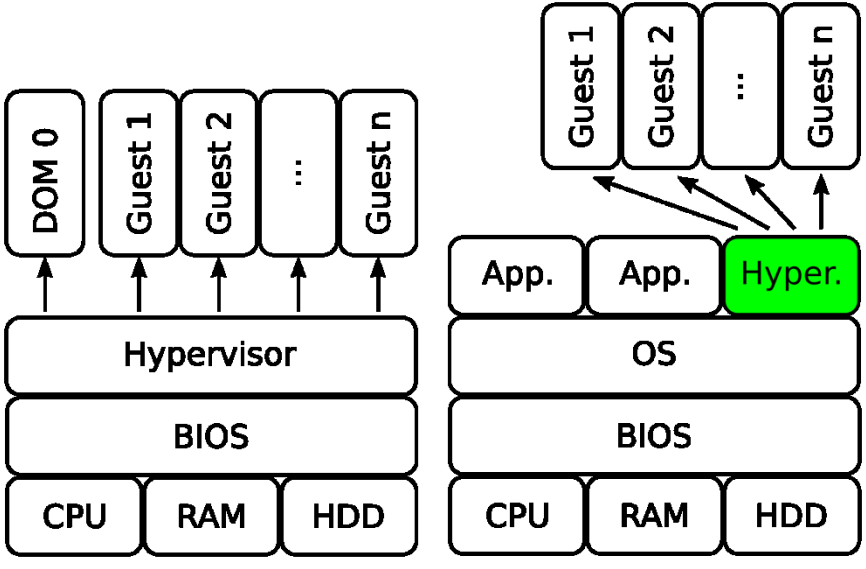
- Predefined setup of external programs, environment variables, libraries and parameters
- Different levels of hardware and system abstraction
- Easy to provide and to share
- Increase reproducibility

VM - Xen [2]



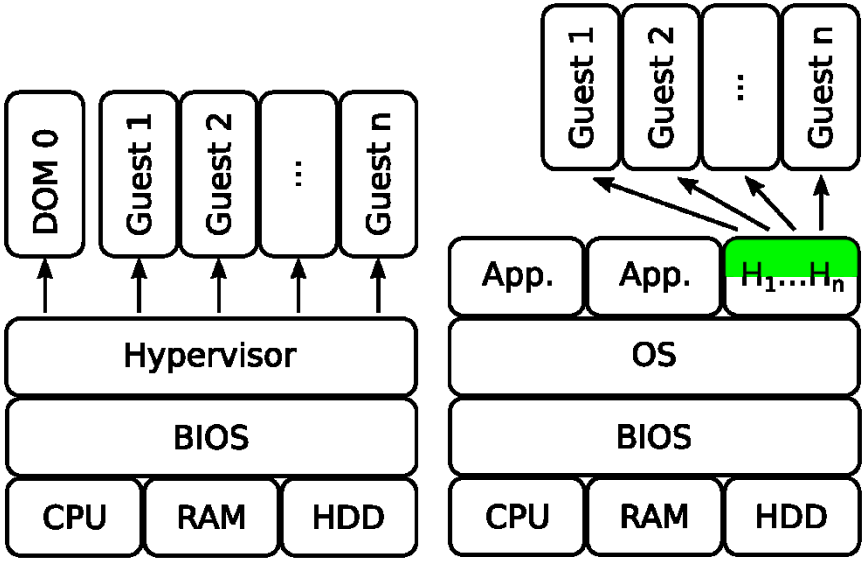
- Hypervisor between hardware and OS
- Slight overhead
- Good isolation of host and guests
- Each guest with own hardware driver possible
- Complex configuration
- Consumer GPU/CUDA limitations

VM - KVM [3]



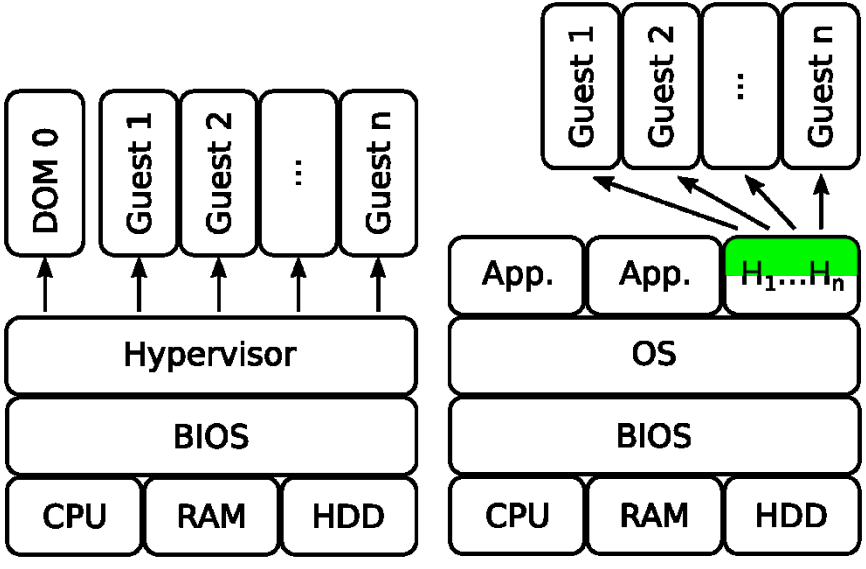
- Hypervisor between host OS and guests
- Slightly more overhead
- Good isolation of host and guests
- Each guest with own paravirtual hardware driver possible
- Easier and flexible configuration
- Consumer GPU/CUDA limitations

VM - Docker (I)[4]



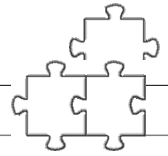
- Hypervisor between host control application and guests processes
- Slightly more overhead
- Isolation of processes, libraries and user environments
- Guests share kernel, drivers and host resources (→ Linux only)
- Simple and flexible configuration

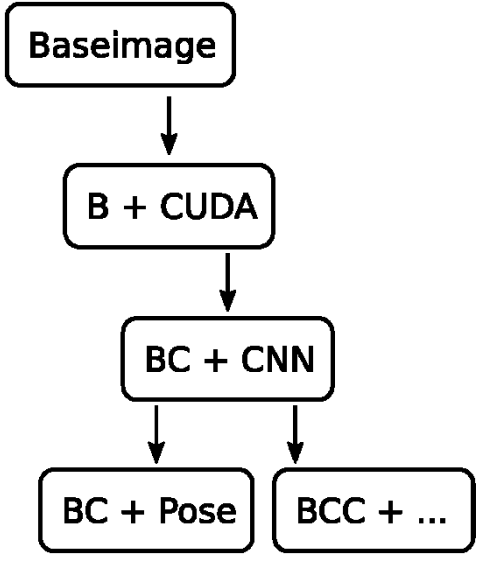
VM - Docker (II)



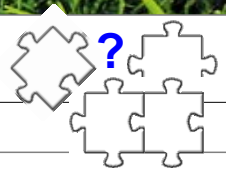
- Guest creation possible with:
  - Version control
  - Inheritance from existing image
  - Easy setup of guest configuration and environment
  - Reproducability
- On Linux, GPU access through host driver on Linux with *nvidia-docker*[5]

VM - Docker (III)






```
FROM nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04
MAINTAINER robert.manthey@informatik.tu-chemnitz.de

# Args
ARG PackageFile="Packages.txt"
ARG CPUNumbers="8"

# Copy data to image
ADD ${PackageFile} /${PackageFile}

# Install common packages
RUN apt-get update && apt-get upgrade -y && apt-get install -y $(cat /${PackageFile})

# Upgrade pip
RUN pip install --upgrade pip

# Install Caffe, Models & OpenPose
RUN cd /opt/; \
    git clone https://github.com/CMU-Perceptual-Computing-Lab/openpose.git; \
    cd /opt/openpose/3rdparty/caffe/; \
    cp Makefile.config.Ubuntu16_cuda8.example Makefile.config; \
    make all -j${CPUNumbers} && make distribute -j${CPUNumbers}; \
    cd /opt/openpose/models/; \
    ./getModels.sh; \
    cd /opt/openpose/; \
    cp ubuntu/Makefile.config.Ubuntu16_cuda8.example Makefile.config; \
    make all -j${CPUNumbers}
# Configs
#RUN

# Create user
RUN useradd -r -U -u 1000 openpose && echo "openpose ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/openpos

# Clean up
RUN rm /${PackageFile}
RUN rm -rf /var/lib/apt/lists/*

# Run when the container launches
WORKDIR /opt
CMD ["/bin/bash"]

## Build
# docker build -t "tuc/openpose" .
```



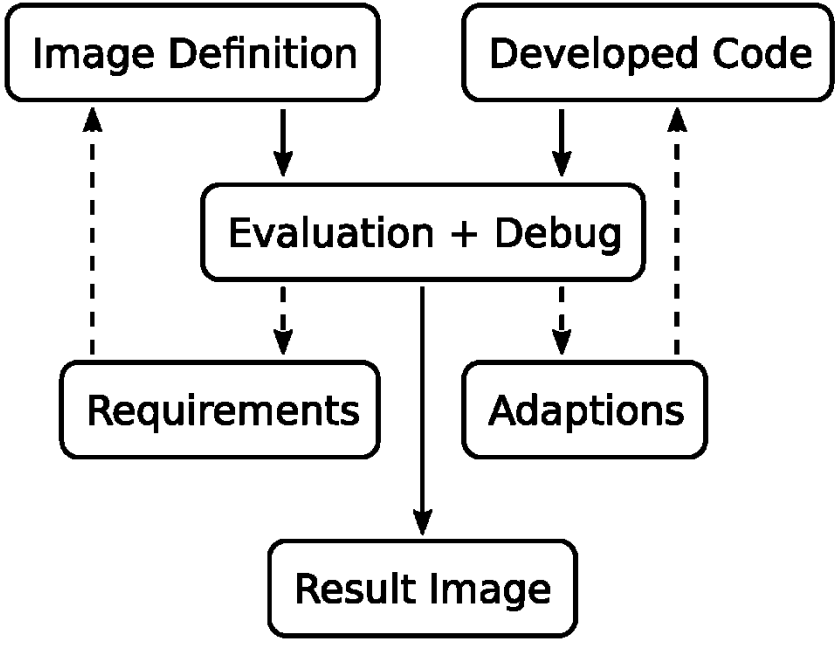
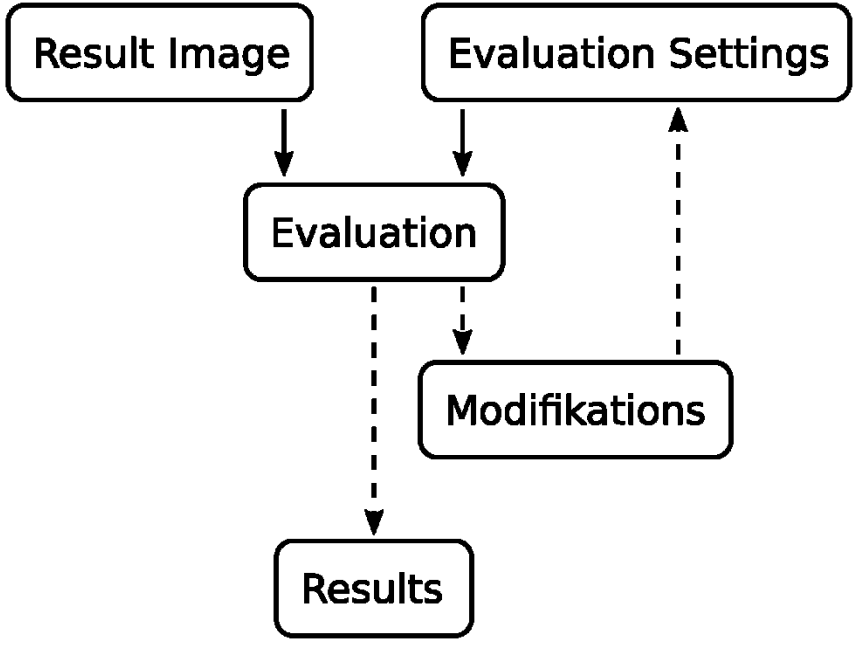


Image Definition
Developed Code
Evaluation + Debug
Requirements
Adaptions
Result Image


Result Image
Evaluation Settings
Evaluation
Modifikations
Results

Example - Openpose [6]

Summary

- Virtual maschines provide different granularity of isolation and abstraction from other guests and hardware (GPU)
- Virtual maschines provide clean, easy to share, reproducable, scalable and fast environments for development, verification and evaluation of algorithm and systems
- Can prevent unneeded debugging

Thank you for your attention.
Any questions?

References

1. https://www.explainxkcd.com/wiki/index.php/1722:_Debugging
2. https://www.xenproject.org/
3. https://www.linux-kvm.org/
4. https://www.docker.com/
5. https://github.com/NVIDIA/nvidia-docker/wiki
6. https://github.com/CMU-Perceptual-Computing-Lab/openpose