GOETHE UNIVERSITÄT FRANKFURT AM MAIN

www.goethe-universitaet.de

Machine Learning II: SS 19

Visvanathan Ramesh

Model-based Design & Simulations for ML

***With contributions from numerous collaborators in over 25 years.**

*Slide source credits: U Washington, Stanford U. (1994/1995), European Conference on Computer Vision 2010 presentation from Siemens AG (publically released industrial perspective), Jian-Bin-Huang and Joerg Bornschein, Patel et al (2015) (deep learning), BFNT-Frankfurt team (2011-2015).

Today’s Class:
Systems Engineering for Vision & Simulation for Modern ML

Summary

System engineering example we will discuss in class:

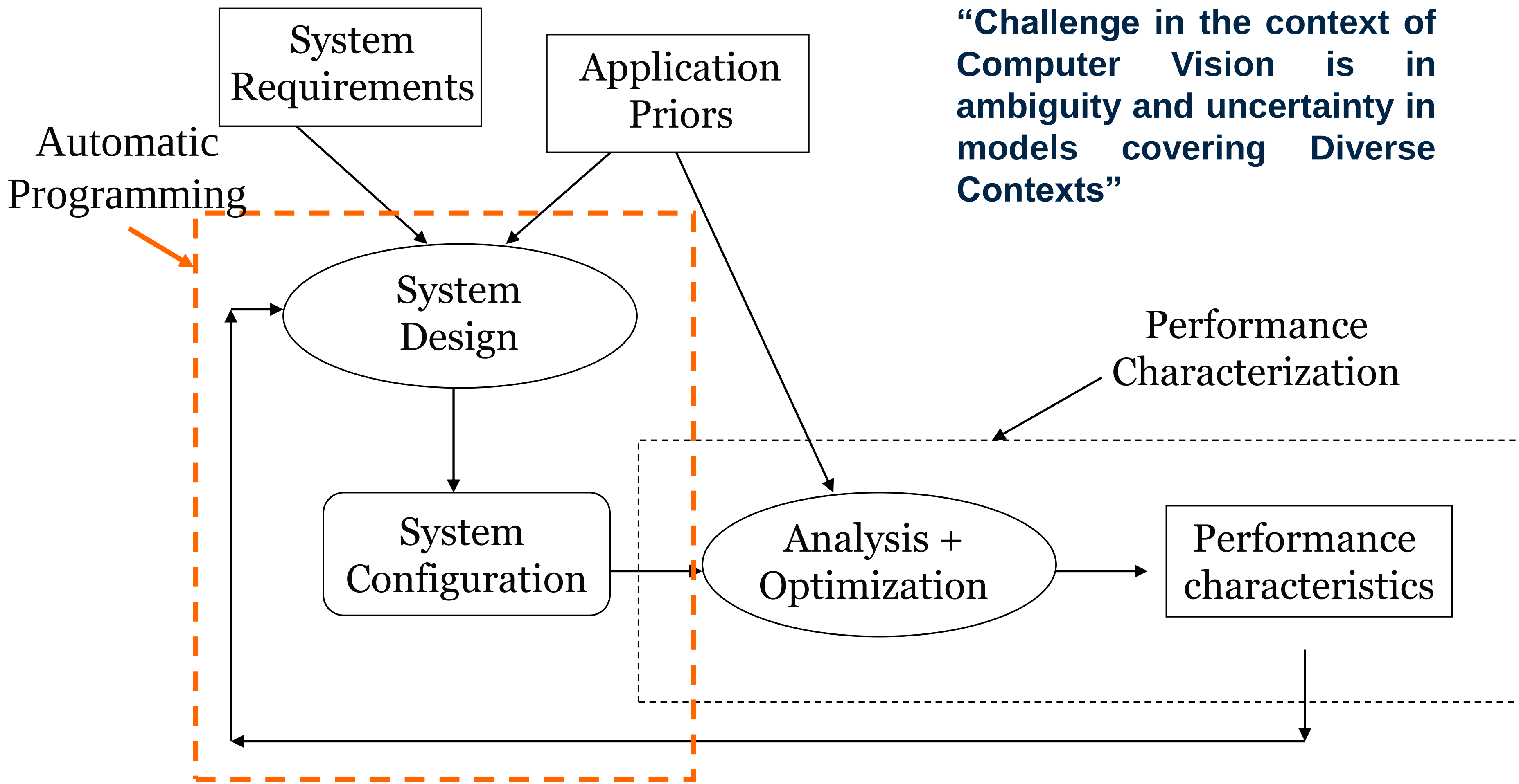
- Greiffenhagen et al (2001)
- Simulation for Design - S. Veerasavarappu et al (2013-17), Hess et al (2016), Weis et al (2015-17)

Overall theme in Model-driven Design:

(Context,Task,Performance) → Hw plus Sw configuration (hw + programs plus parameters)

- Context, Task, Performance
  - What is context – e.g. Derek hoeim's Book (2015)
  - Task - estimation of world state (or aspects of it)
  - Performance - bias, variance , accuracy vs speed tradeoff
- What is a Program (Inference Engine)?
  - Program as filters and combinations (feedforward, deep, feedback and recurrent) (ML Literature, Bio-inspired vision literature)
  - Program Design – Model based vs Data Driven, or Hybrid combinations
  - Classic dissertations: Model based design - (Mann, 1996 + Ramesh, 1995)
    - Graphical model illustration using VSCP
    - Inference Aggregation Indexing , prediction, verify loops , Hierarchical
    - What about Performance Characterization? (Ramesh, 1995)

Systems Design & Analysis Cycle



"Challenge in the context of Computer Vision is in ambiguity and uncertainty in models covering Diverse Contexts"

Requirements Specification for Real-time Vision Systems

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Input Space specification:

- Object-oriented Graphical Models describing generative models for video data given scene variables
- Scene variables include:
  - Scene Geometry (static geometry), Material distribution, Environmental Conditions (e.g. weather, indoor, outdoor), Object types in the scene, their shape, dynamics, Illumination distribution (e.g. source positions, dynamics), Camera (Sensor) positions, orientations in the world, projection geometry, photometric model

Task Specification:

- Desired subset of scene parameters to be estimated from video (for example):
  - Counts of object
  - Object types, Object tracks, Object geometry, Object behavior
  - Analysis of Groups of objects
  - Illumination/weather state

Performance Requirements:

- For each task: probability of error (e.g. p_miss, p_false in two class situations)
- Accuracy in Parameter estimates (tolerances)
- Graceful degradation, Self-Diagnosis
- Computational speed
- Time delay to respond (i.e for computation of results), etc.

Desired Properties of Vision System Designs

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- **Modularity in Specifications:**
  - Nested model spaces to allow for various degrees of approximations in the model space
- **Scalability of Design Solutions:**
  - Ability to derive families of solutions where the complexity of system is scaled according to complexity of tasks, input space approximations.
- **Quantifiability:**
  - Ability to provide quantitative performance models of system designed as a function of Graphical Model parameters and tuning parameters/constants of system.
- **Computational Complexity tradeoff vs Accuracy:**
  - Ability to quantify computational complexity of system as function of OODBN parameters.
  - Use this quantification to provide tradeoffs (e.g.) Reduce accuracy for reducing computation.
- **Modular Extensibility:**
  - Design should allow for modular extensibility when input spaces in one application differ from another in a minor way.
- **Mapping to Hardware:**
  - Design should allow ease of mapping to target hw. (could address this as a separate phase. (i.e). Construct designs for general purpose architectures and then have a systematic approach to translation of design to hw.

Paradigms for Design

- Model Based Design
  - Generative Models – i.e. Probabilistic Graphical Models (Interpretation is estimation of world state given observations. Generative model uses a likelihood model for sensor observations (physics-based) and Prior model.)
- Data Driven Machine Learning
  - Neural Networks
  - Boosting, Support Vector Machines, etc.
- Hybrid designs (combination)

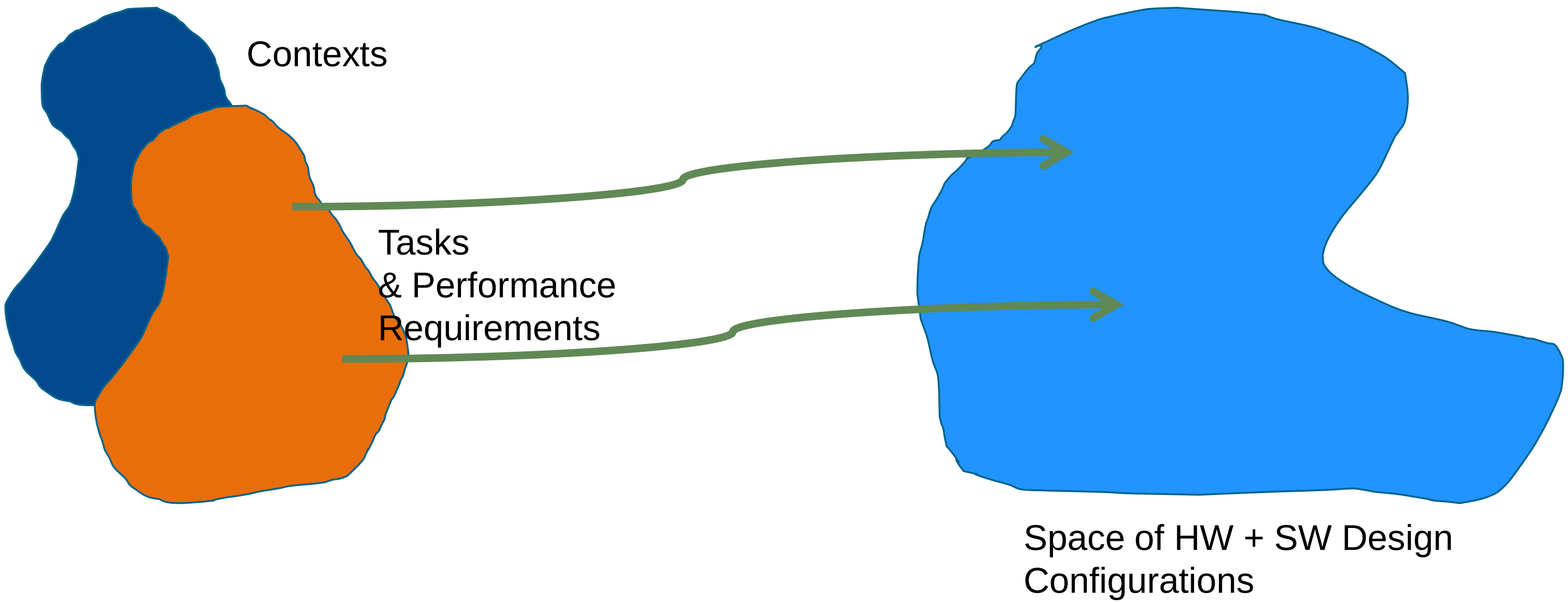
Systems Engineering: Key Ideas

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- Formalize domain (i.e. generative) models for application contexts
- Formalize system task requirement specification
- Translate requirements to formal generative models
- Link generative models to approximate inference engines (i.e. module and system implementations)
- Performance characterization of design (white box analysis)
- Model Validation and Iteration of Design (comparison of empirical and theoretical predictions and model/design improvement)

Key Insight: Learning of (C, T, P) → Program mappings

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

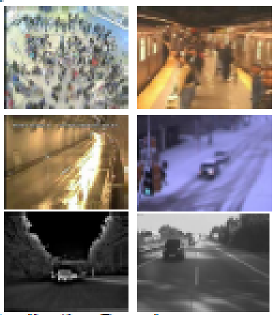


"(Contexts, Task and Performance Requirements) → to (System Designs)"
Extension to new design settings – via re-use of context elements and identification of gaps in models

Methodology Summary

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

User View Of System



Application Domains:
- Automated Video Surveillance for Safety and Security
- Vision for Autonomous driving – comfort and safety

Task Specification:

e.g. A situational aware system that: parses video from indoor and outdoor environments and generates descriptions of objects, events and is able to alert to a user when event descriptions match user specified queries.

Performance Specs:

(Functional as well as non-functional):

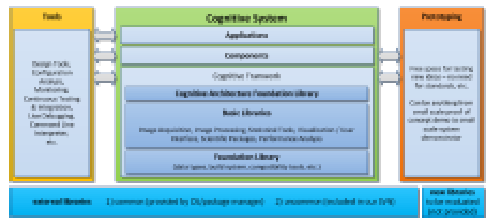
For each task: probability of error (e.g. probability of miss detection, probability of false alarm in two class situations)
- Accuracy in Parameter estimates (tolerances)
- Graceful degradation, Self-Diagnosis
- Cost, Computational speed
- Time delay to response, etc.

Modeling View of System

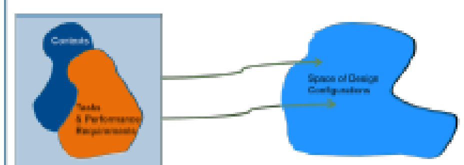
Formal Graphical Models of Context (of various degrees of approximation) along with Physics based Sensor models are linked to systems level analysis of sequential estimators.

Overview: A vision system is complex and composed of multiple stages of execution and is a non-linear estimator from Sensed Images → Semantics. Choice of estimator depends on application requirements and graphical models and involves approximations. Core ideas use in the methodology include:
- Statistical methods for describing population of inputs, quantifying performance and limitations, and to perform inference.
- Use of Probabilistic Contextual Models - Represent application constraints in terms of: priors on camera parameters and projection model, object parameters and dynamics, priors on illumination.
- Develop likelihood models L(features|image) taking into account sensor and feedforward computational process models correctly. Perform Bayesian inference by combining likelihood model and priors.
- Understanding of Vision application space → likelihood model choices and prior choices leads to the building of systems in a rapid and cost effective way.

Implementation View of System

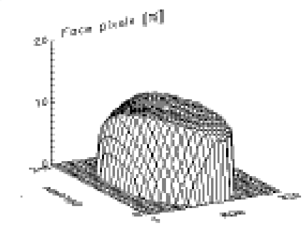


Cognitive Vision Platform for Rapid Prototyping & Validation



"Explore Model Space for Contexts, Task and Performance Requirements, Systematically Map them to Design configurations, Prototype Designs, Validate, Iterate" as in 'Design Thinking'. Extension to new design settings – via re-use of context elements and identification of gaps in models."

Evaluation & Validation of System

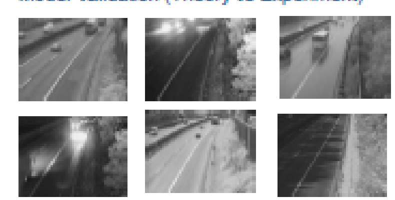


System Performance Model (Example)

Validation. First two lines show the predicted (hat), and experimental (tilde) variances for the tilt angle α. Next two lines correspond to pan angle β.

$\times 10^{-3}$	P1	P2	P3	P4	P5	P6	P7	P8
$\hat{\sigma}^2_{\tan\alpha}$	2.10	2.12	1.57	1.40	1.35	1.31	1.31	1.32
$\tilde{\sigma}^2_{\tan\alpha}$	2.05	2.04	1.60	1.34	1.36	1.32	1.40	1.31
$\hat{\sigma}^2_{\sin\beta}$	28.9	26.1	21.8	17.9	15.8	15.3	18.4	20.1
$\tilde{\sigma}^2_{\sin\beta}$	25.9	24.1	19.5	15.1	14.9	15.0	18.1	19.3

Model Validation (Theory vs Experiment)

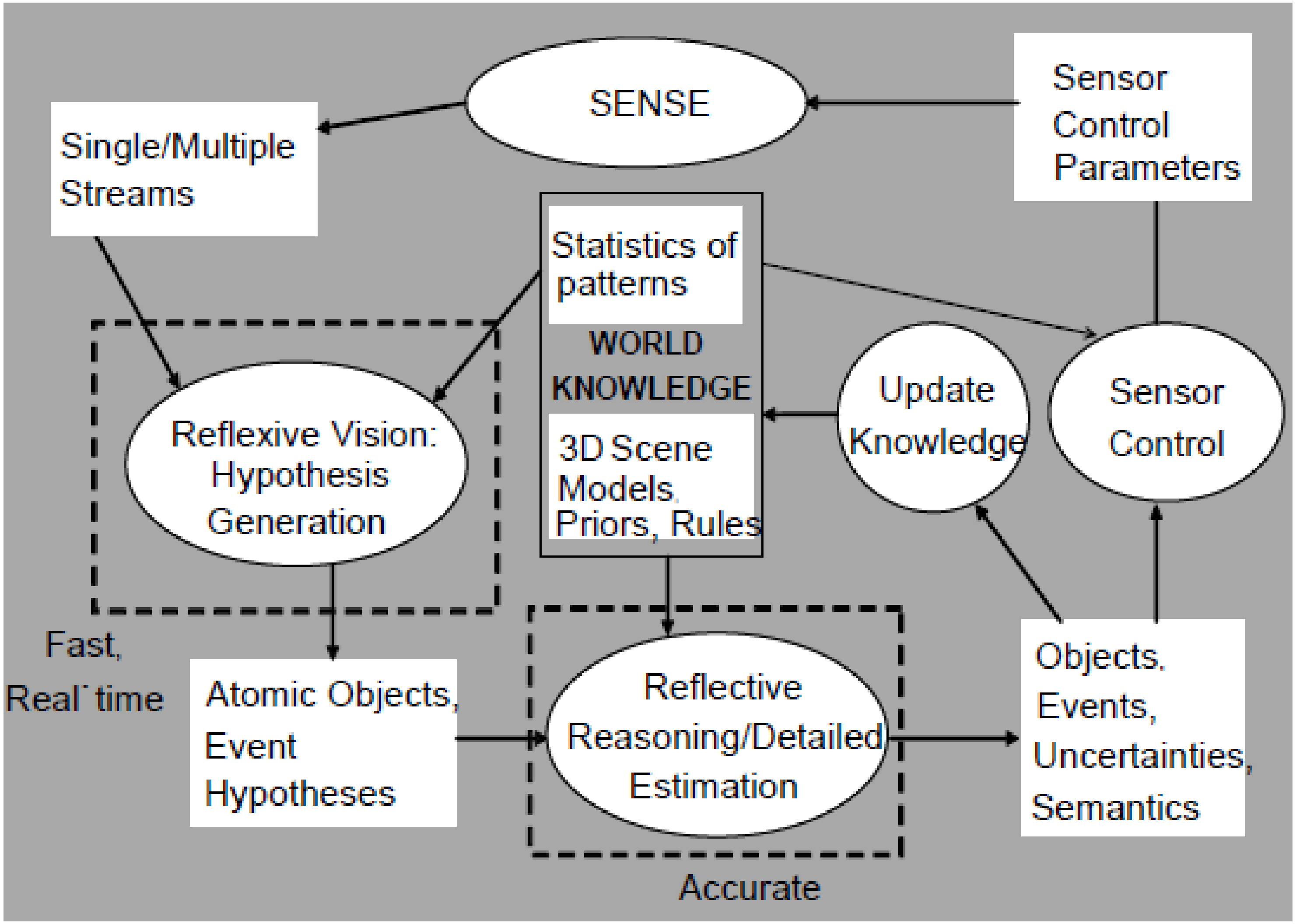


Large scale Field Trials & Validation

"Visual Cognition is 'quasi-invariant Indexing' followed by detailed estimation (or deliberation, iteration) – component level research is quite mature, open research is on systems questions involving Cognitive Vision Platform with Continuous Learning and Self-Diagnostics".

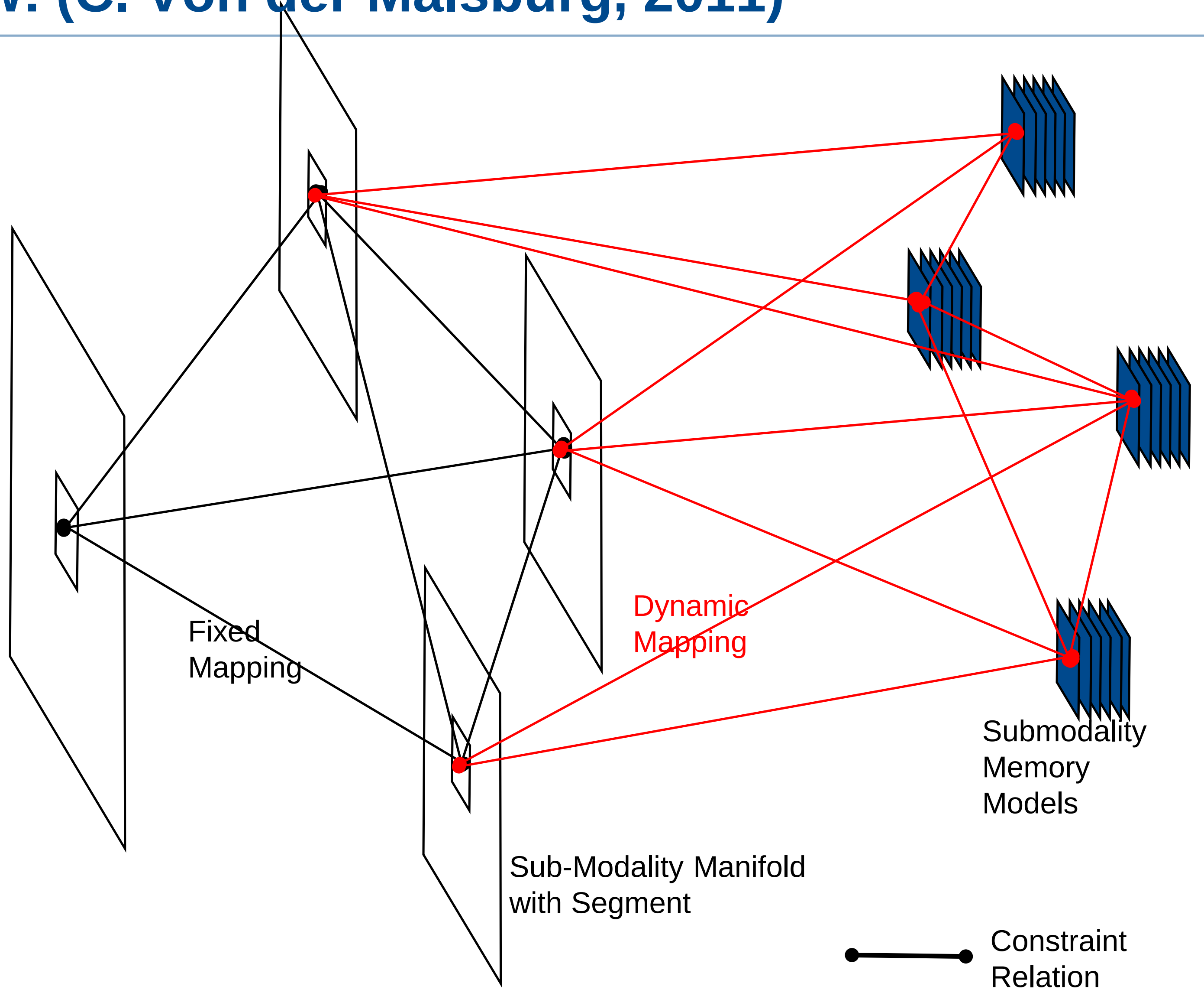
Essence of Overall Design Framework: {Application contexts} x {sensor types + configurations} x {questions posed} x {perf specs/requirements} ----> {specific hypotheses generators} + {reasoning / optimization engine}

Visual Cognition: Hierarchical Indexing + Iterative Estimation

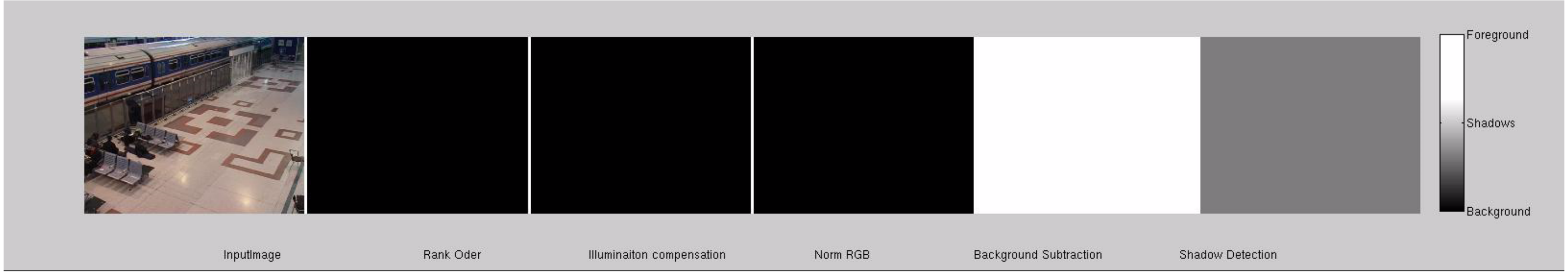


Modules
Sense
Hypotheses Generation
Reasoning/ Optimization
Fusion Strategy
Online Learning
Control & Action
Memory Representation

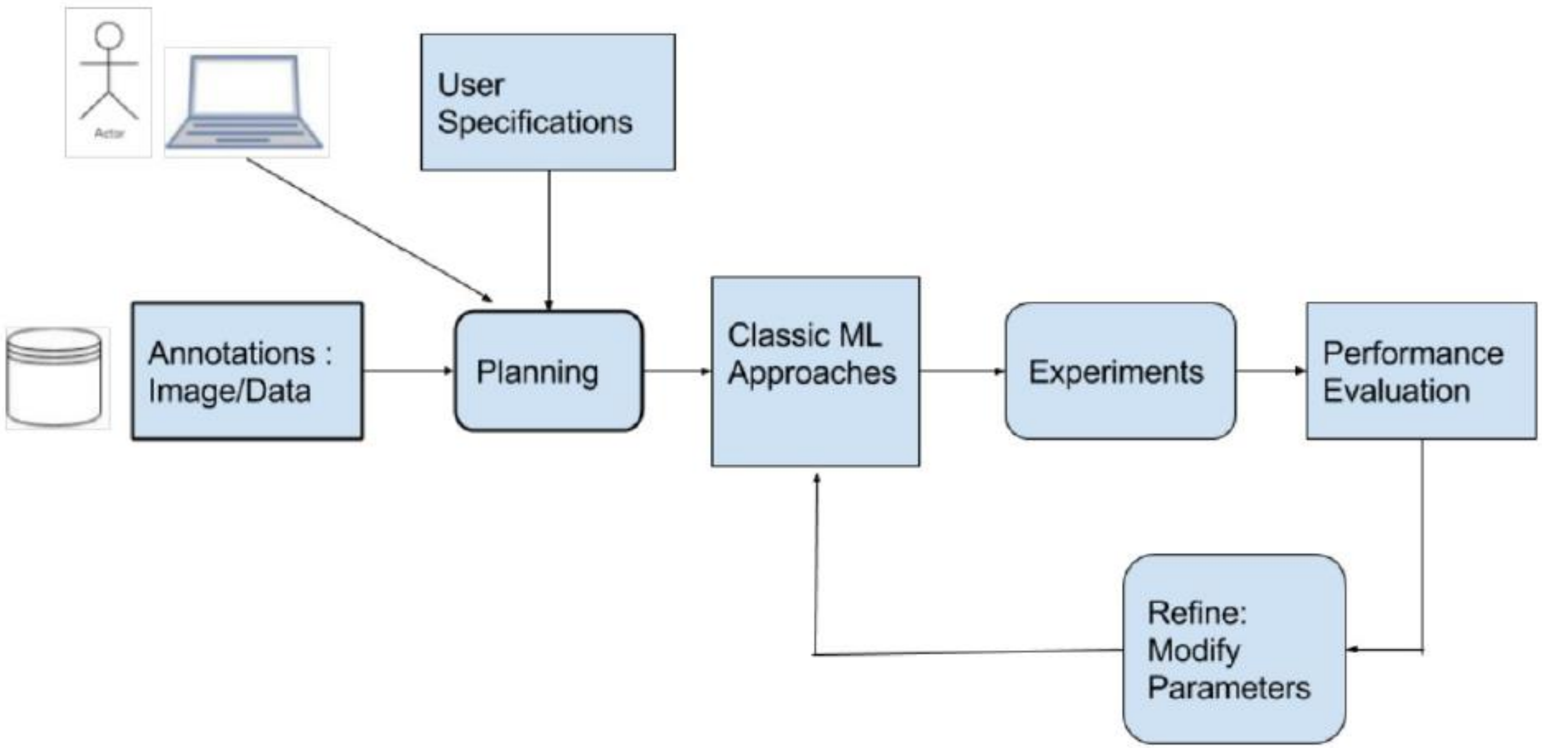
Computational Neuroscientist's View: (C. Von der Malsburg, 2011)



Demo Video Illustrating Decomposition



System Design Process



Design Work flow – From Skeleton Designs to performance evaluation

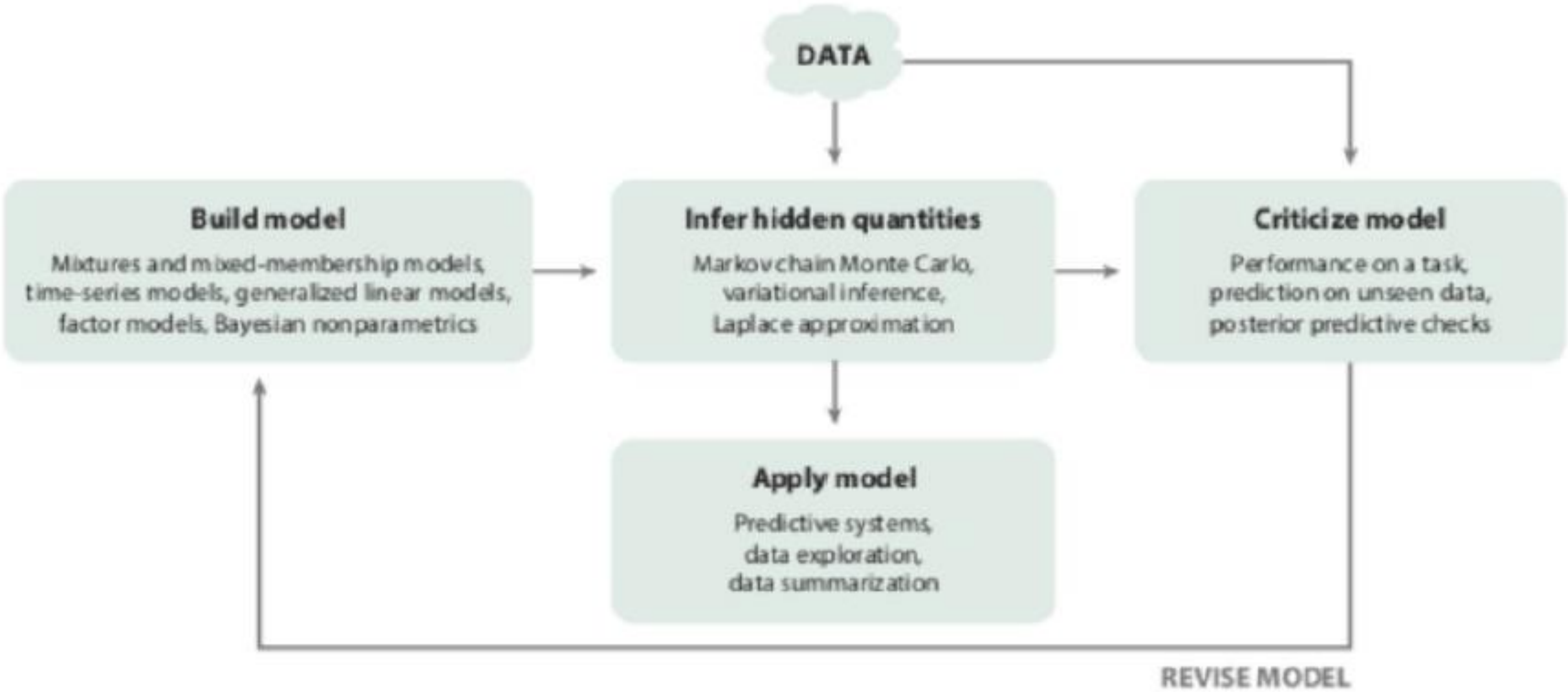
Solution Approaches

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- Model Based Design
- Data Driven Design
- Hybrid Approach : Considering both model and data driven designs

Solution Approaches

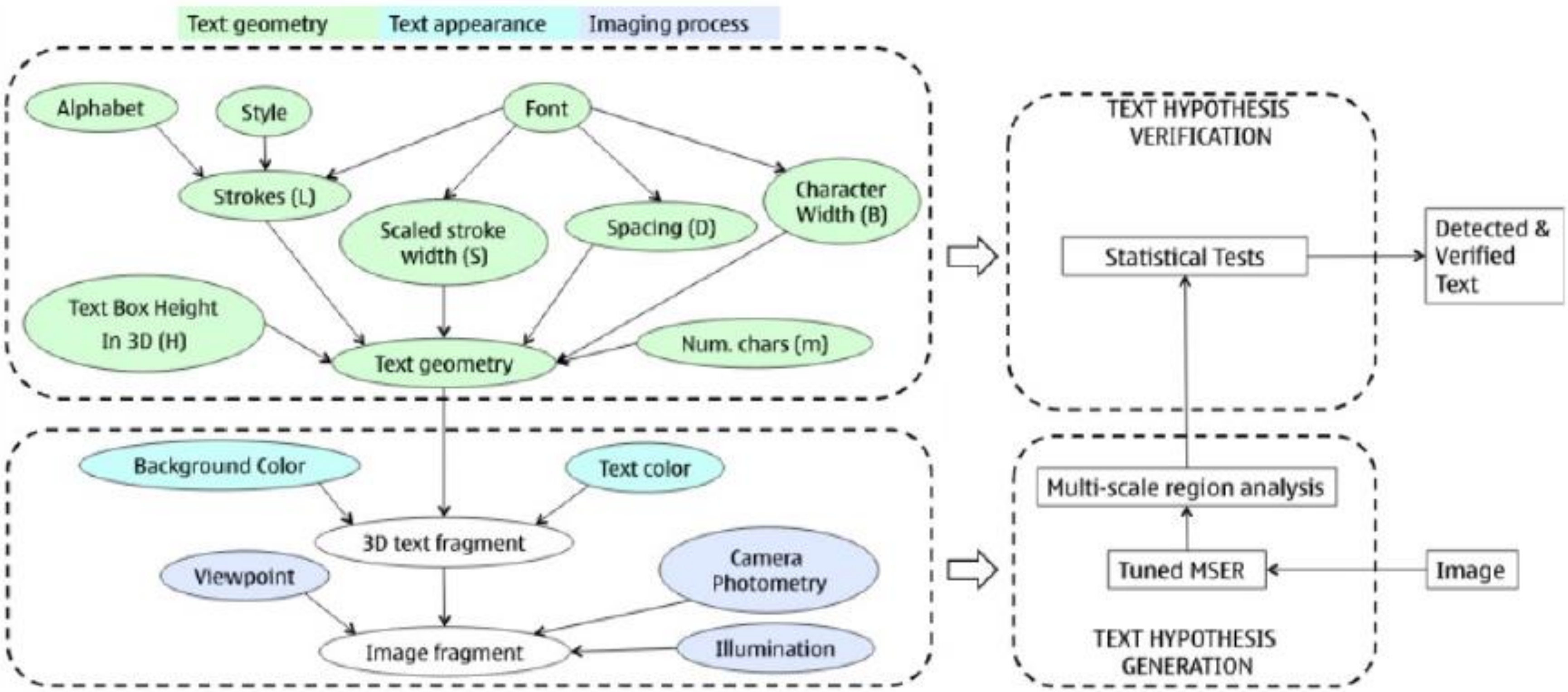
1. Model Based Design



General setup of Model based methods.
Image Source: [6], Blei (2015)

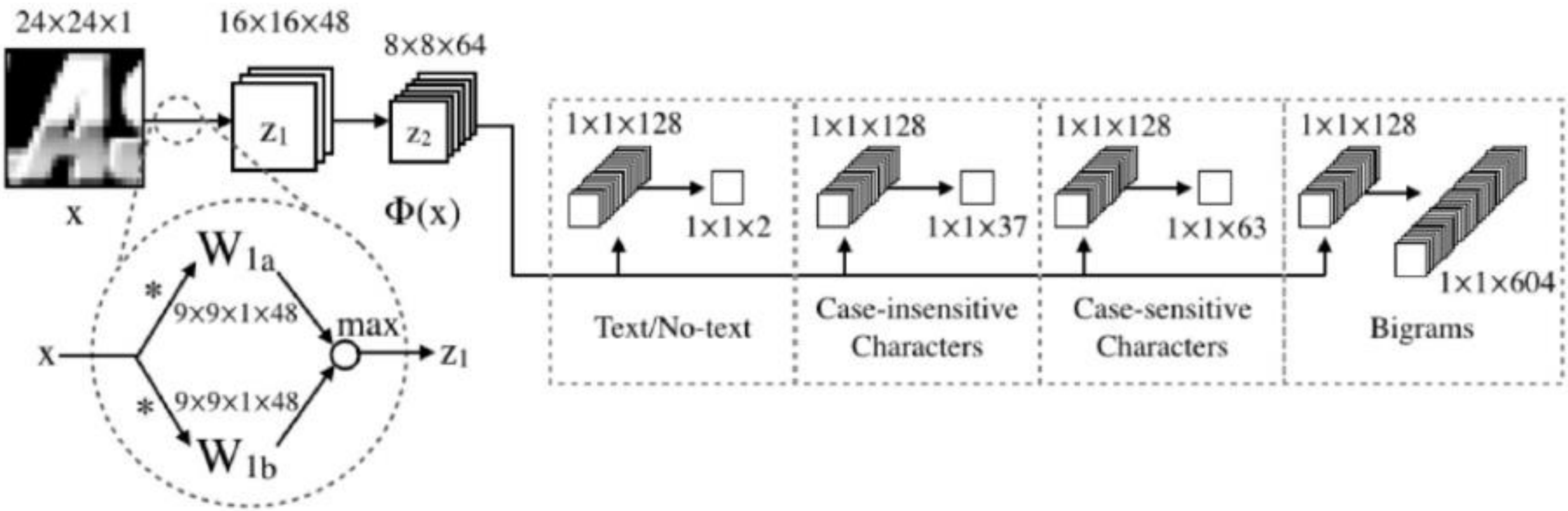
Solution Approaches

Classic Example for Model Based Design



Left: Bayesian Network for Text Appearance in an Image.
System Design (See [3])

2. Data Driven Design



Convolutional Neural Networks. The method uses four CNNs. These share the first two layers, computing "generic" character features and terminate in layers specialized into text/no-text classification, case-insensitive and case-sensitive character classification, and bigram classification. Each connection between feature maps consists of convolutions with maxout groups. Figure and caption from [7]

Solution Approach: Hybrid

- Combine strengths of model-based thinking as well as data driven machine learning.
- Several feature maps are extracted based on several feature extraction kernels.
- This is followed by a deep neural network architecture or any data driven architecture for the purpose of classification and recognition.

Simulating Worlds

for machine learning

Authors: Tobias Weis, Timm Hess, Subbu Veerasavarappu and Visvanathan Ramesh

Contextual Models: Simulation from Generative Models

- Camera geometry -- projection model (orthographic, perspective), camera blur, lens distortion, intrinsic parameters, extrinsic parameters.
- Camera - gray level transformation model of camera pipeline
- Shape representation (surface/contour, volume)
- Material property (brdf)
- Appearance (texture map) dictionary
- Graphics pipeline parameters

Simulation for Systems Design, Analysis and Evaluation

- **Groundtruth collection seems to be an obstacle for Supervised learning based vision systems.**

- **Major advances in Computer Graphics (CG) field has spurred a renewed recent interest to utilize CG for CV.**

(a) 2001 (b) 2003 (c) 2005

(d) 2006 (e) 2013 (f) 2015

Figure 1-1: Evolution of Graphics in Video games from 2001 to 2016

Rendered Data - various scene conditions


Lambertian


Ray traced


Path traced (130 spp)

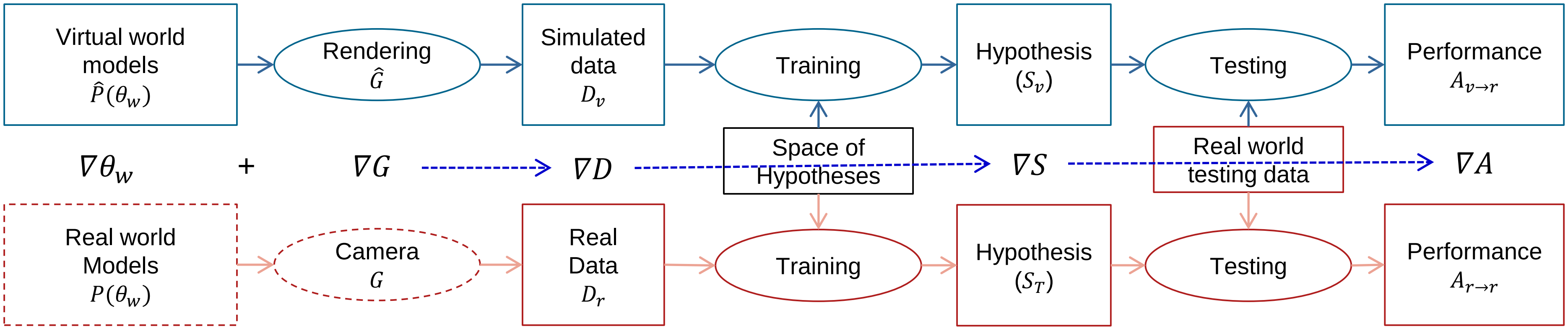

Noon


Night


Rain

Annotations are “free”

Transfer and Domain shift



- No free lunch in the selection of $\hat{P}$ and $\hat{G}$ for data simulation processes.
- In principle, $\nabla\theta_w$ and ∇G impact the magnitudes of ∇D, ∇S, and ∇A.

- ***What is the impact of $\hat{G}$ on ΔA?***
  - Real time Photo-realism vs Expensive physics-realism?
- ***What is the impact of parameters of $\hat{P}(\boldsymbol{\theta_w})$ on ΔA?***
  - How far can we go with an arbitrary scene generative model?
  - Can unsupervised generative learning from target real data help?

- **However, one can bypass these issues by simply adding some real samples to simulated data.**

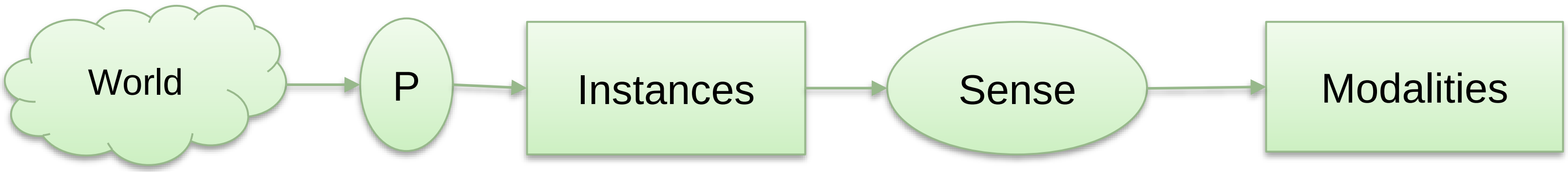
Simulation for ML

Why, when, and how to simulate data

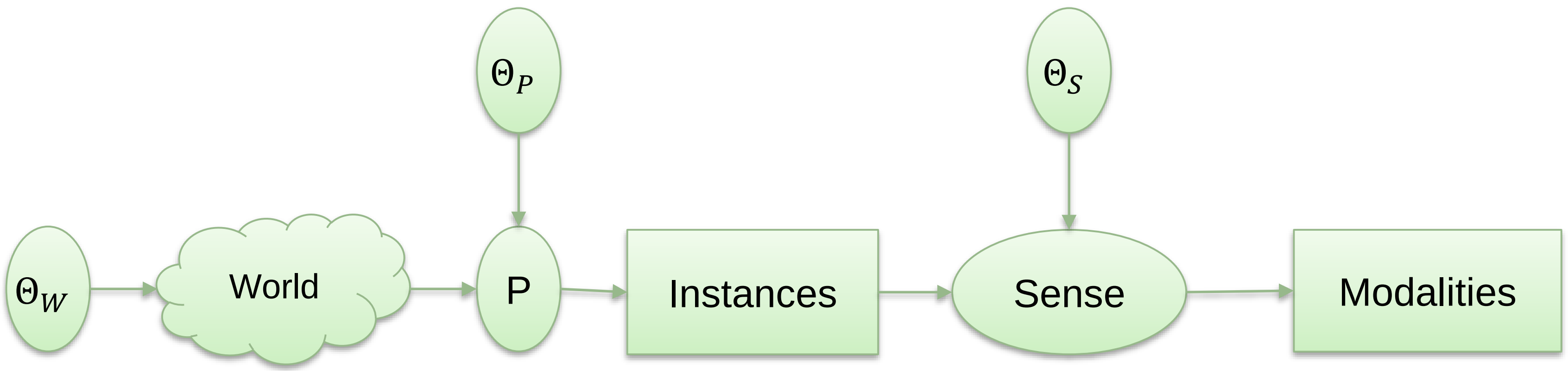
Simulation Software Workflow

Learning from Virtual Worlds

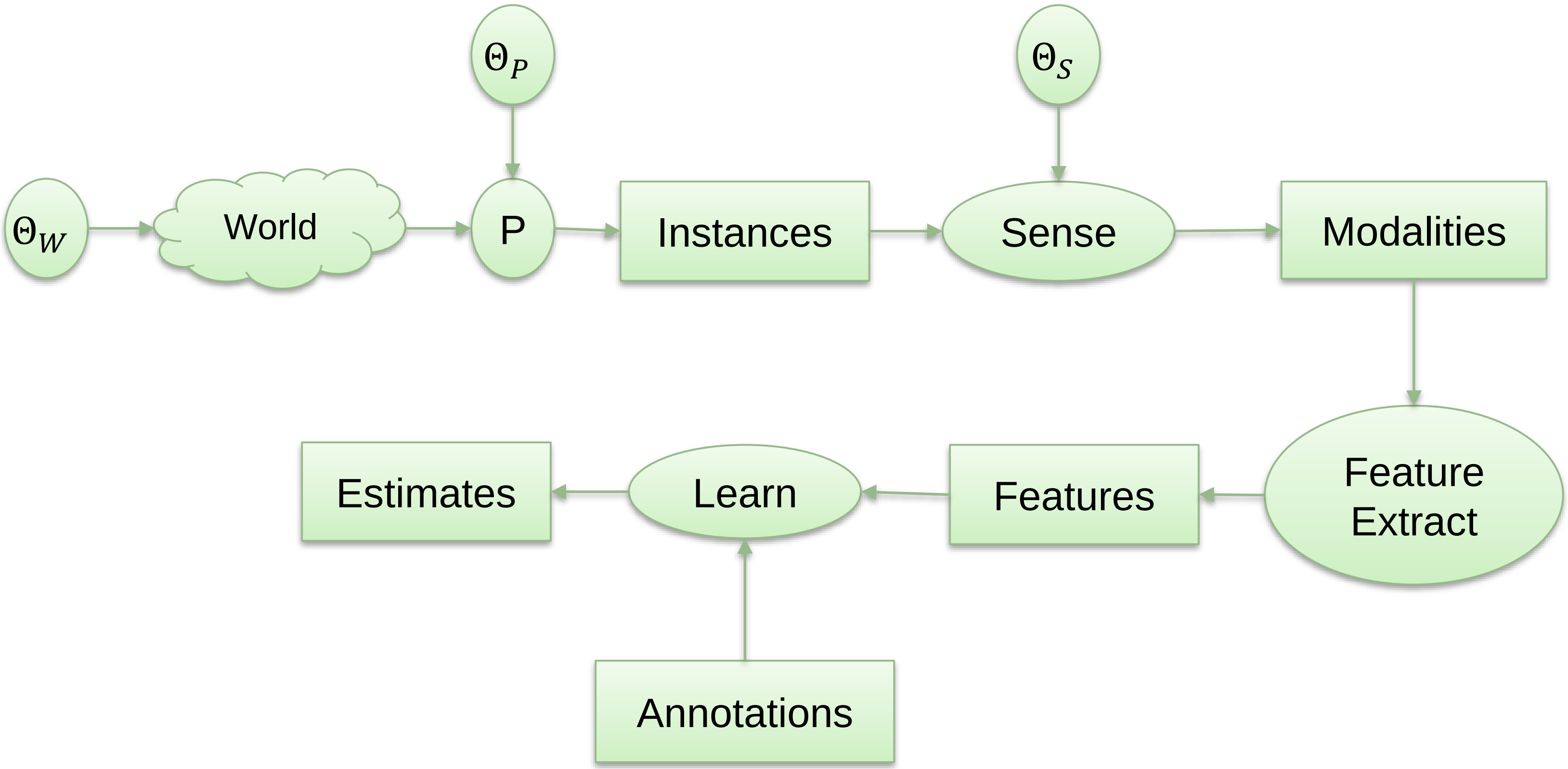
Overview ML Workflows



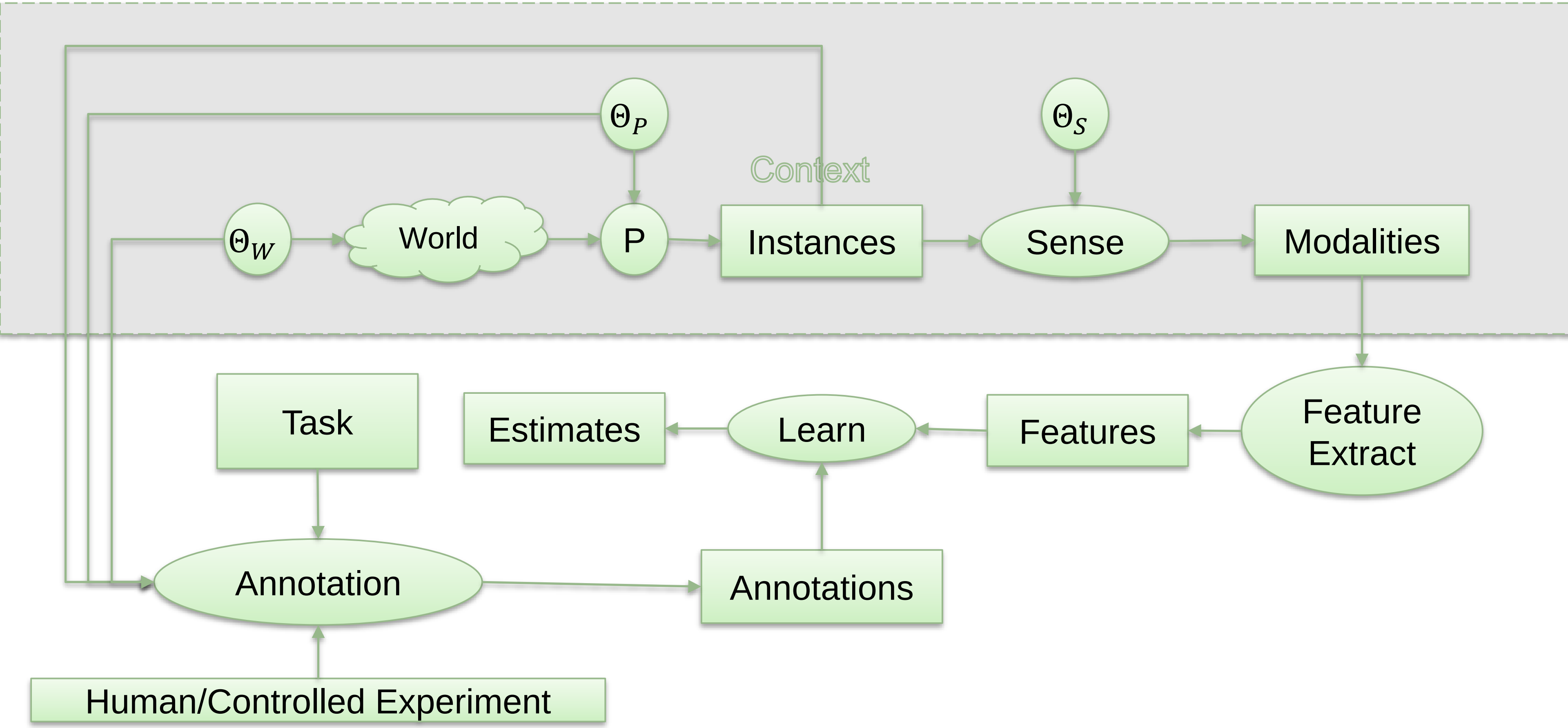
Overview ML Workflows



Overview ML Workflows



Overview ML workflows



Why Simulation

Human annotation is expensive, time-consuming, may contain errors

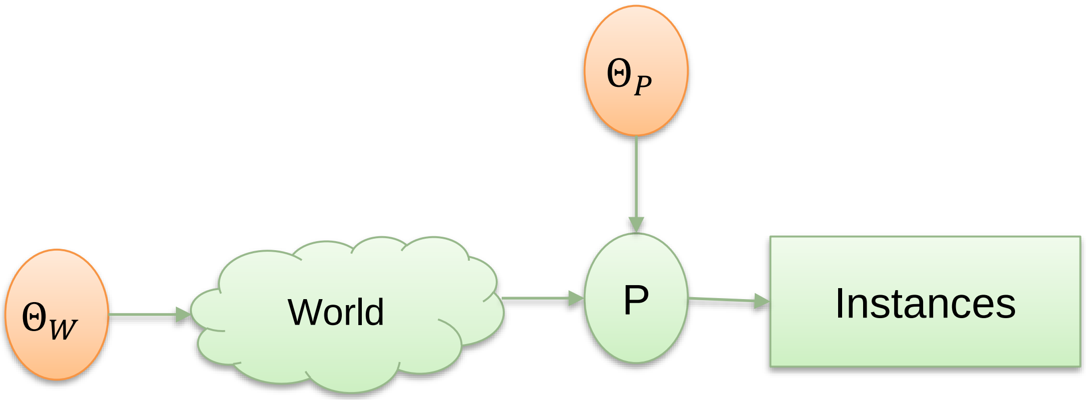
Controlled experiments may be infeasible

Sensed/captured instances may only be a subset of probable situations/scenarios

Simulation Design

GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

The world we sense is governed by parameters:



ML wants to estimate a subset of those: Task

This subset may be influenced/modulated by other variables: Context

Simulation Design

Θ_W: **``world"-parameters:**

Geometric

- Object positions, 3d-shapes, spatial relations

Photometric

- Materials and reflectances, scene lighting, atmospheric effects

Simulation Design

Θ_P: ``process“-parameters:

An instance of the world is a specific configuration of entities, arranged according to a process P

It‘s parameters Θ_P might include physical, social, or other laws that are imposed on entities

Simulation Design

Θ_S: **``sensing"-parameters:**

Describe sensor-specific parameters

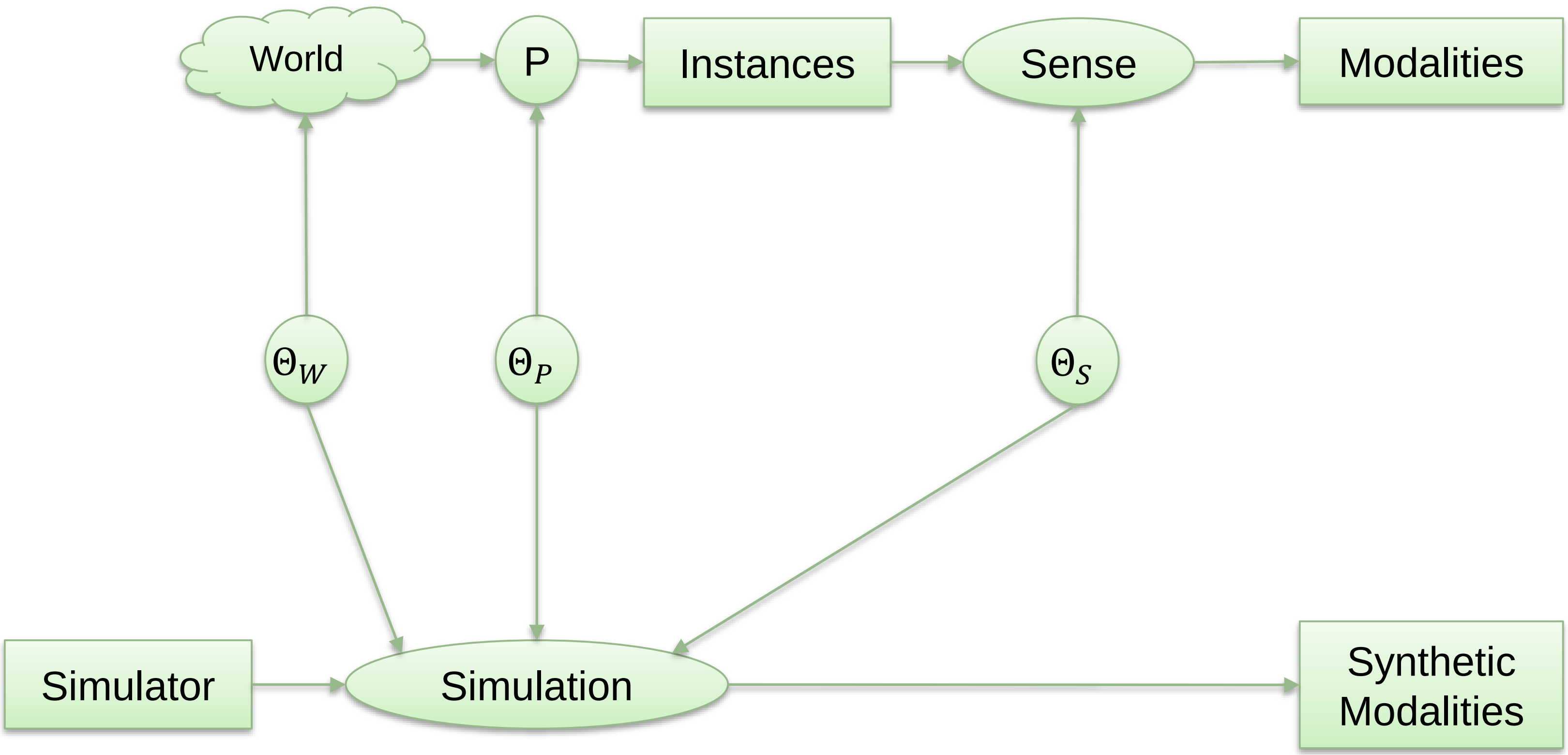
- Transfer-function of physical entities to digital ones (i.e. photon->pixel-value)
- Noise
- Range
- Other characteristics (i.e. lense distortions)

Simulation Design

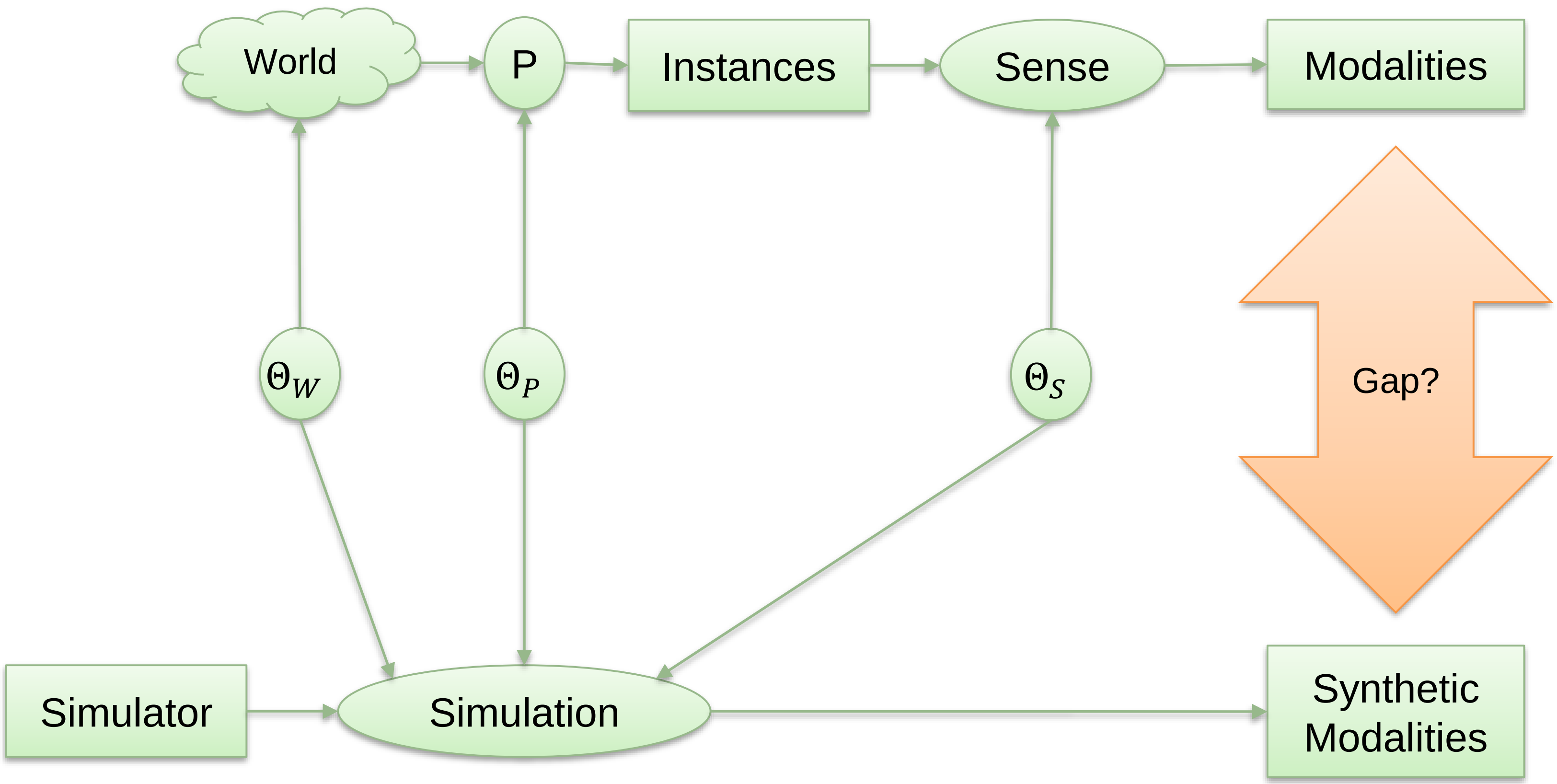
To generate useful synthetic data, we have to:

- **Model parameters of interest (task)**
- **Model influencing parameters (context)**
- **Be able to synthesize needed modalities (sensor)**
- **Be able to generate annotations**

Simulation Design



Simulation Design

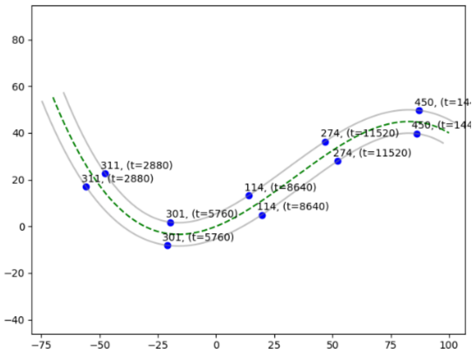


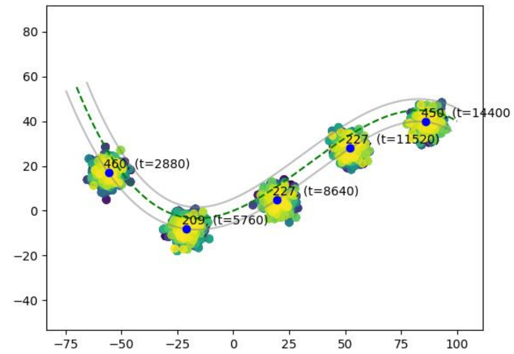
Why build Simulations?

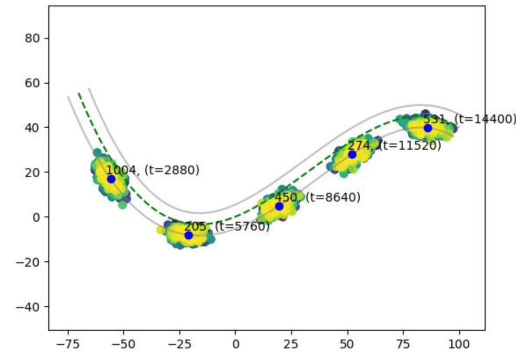
In principle, any quantity of interest can be simulated

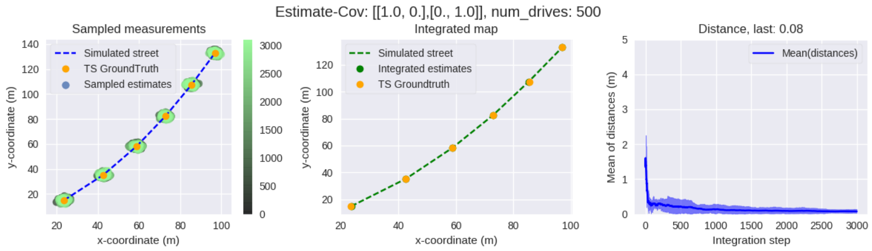
Today‘s session is focused on simulation for Computer-Vision related tasks

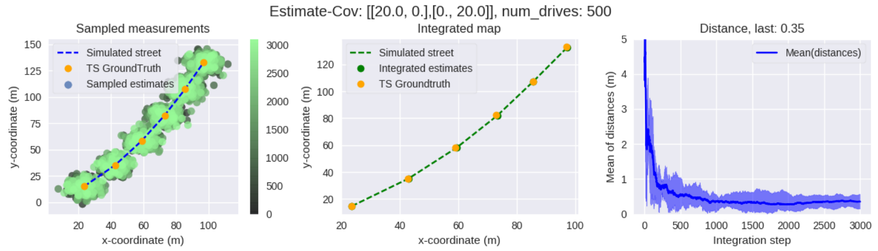
Why build Simulations?











Traffic sign map integration

Task

- Integrate noisy measurements (GPS-Coordinates) into a map

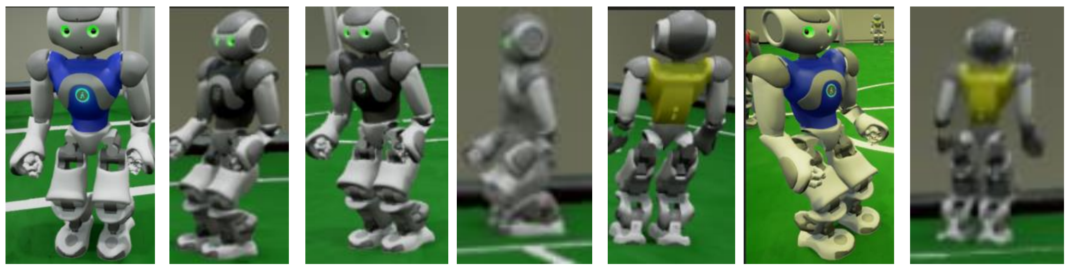
Parameters

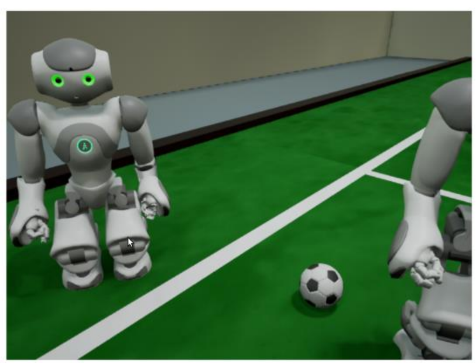
- Θ_W
  - Geometric: CAD-desc of streets, GPS-coordinates of TS
- Θ_S
  - Sensor uncertainty

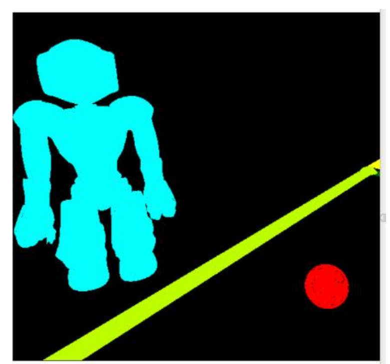
Why build Simulations?

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Possible source of (unlimited) situation examples – Photorealistic RGB images







Examples
RoboCup

Task

- Object (ball, robot, line, background) classification from image patches

Parameters

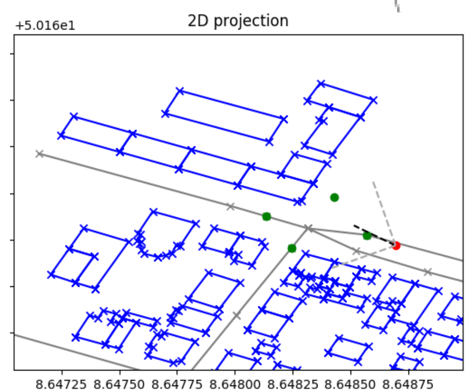
- Θ_W
  - Geometric: CAD-desc of ball, robot, playing field
  - Photometric: textures, reflectances, external lighting (context)
- Θ_P
  - Objects located inside field-border
  - Governed by gravity (everything on ground-plane)
  - Robots are articulated
- Θ_S
  - Sensor resolution, possible exposure times, noise characteristics
  - Extrinsic (position,angle) and intrinsic (focal length, central point, distortion) parameters

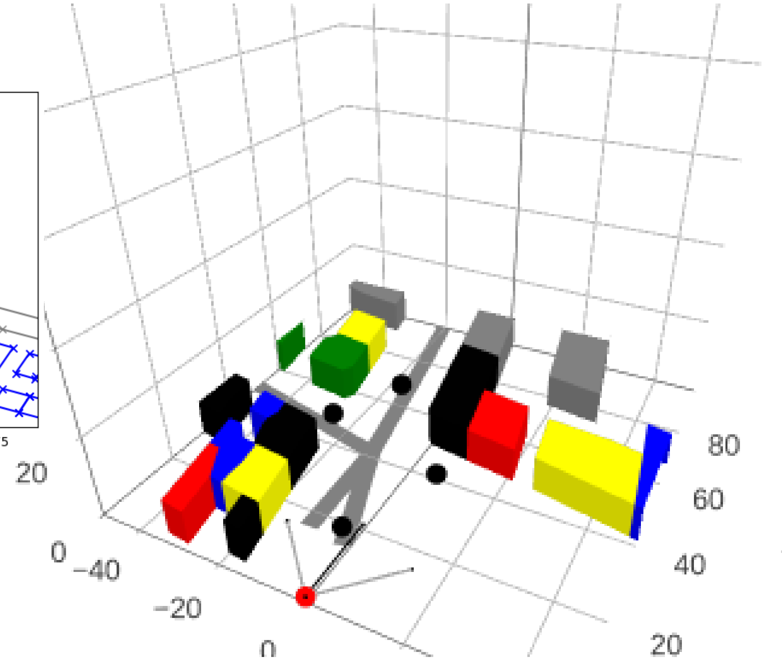
Why build Simulations?

Possible source of (unlimited) situation examples – Optical flow

Why build Simulations?

Possible source of (unlimited) situation examples – 2D/3D Simulations from map data





Examples
Brakelight

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Task

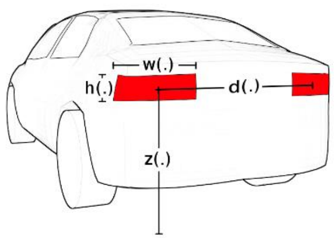
- Detect motion anomalies -> simulate ``normal“ image motions

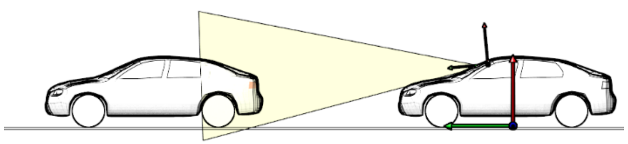
Parameters

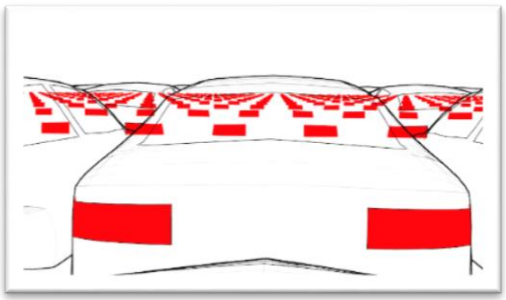
- Θ_W
  - Geometric: CAD-desc of automotive scenes (cars, streets, buildings, etc.)
  - Photometric: textures, reflectances, external lighting (context)
- Θ_P
  - Object locations (buildings on side of street, cars on street, etc.)
  - Governed by gravity (everything on ground-plane)
  - Object motions (other cars, people)
- Θ_P
  - Sensor resolution, possible exposure times, noise characteristics
  - Extrinsic (position,angle) and intrinsic (focal length, central point, distortion) parameters

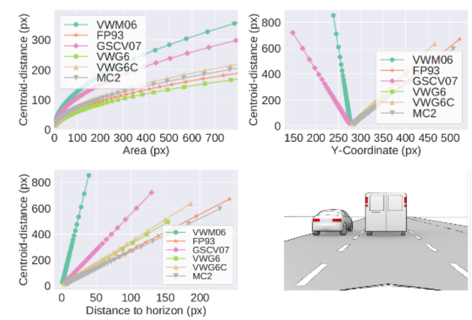
Why build Simulations?

Possible source of (unlimited) situation examples – 2D projections









Brakelight

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Task

- Calculate likelihood of detected blob-pairs (only geometric)

Parameters

- Θ_W
  - Geometric: CAD-desc of cars
- Θ_P
  - Object locations (other cars)
  - Governed by gravity (everything on ground-plane)
- Θ_S
  - Sensor resolution
  - Extrinsic (position,angle) and intrinsic (focal length, central point, distortion) parameters

Why build Simulations?

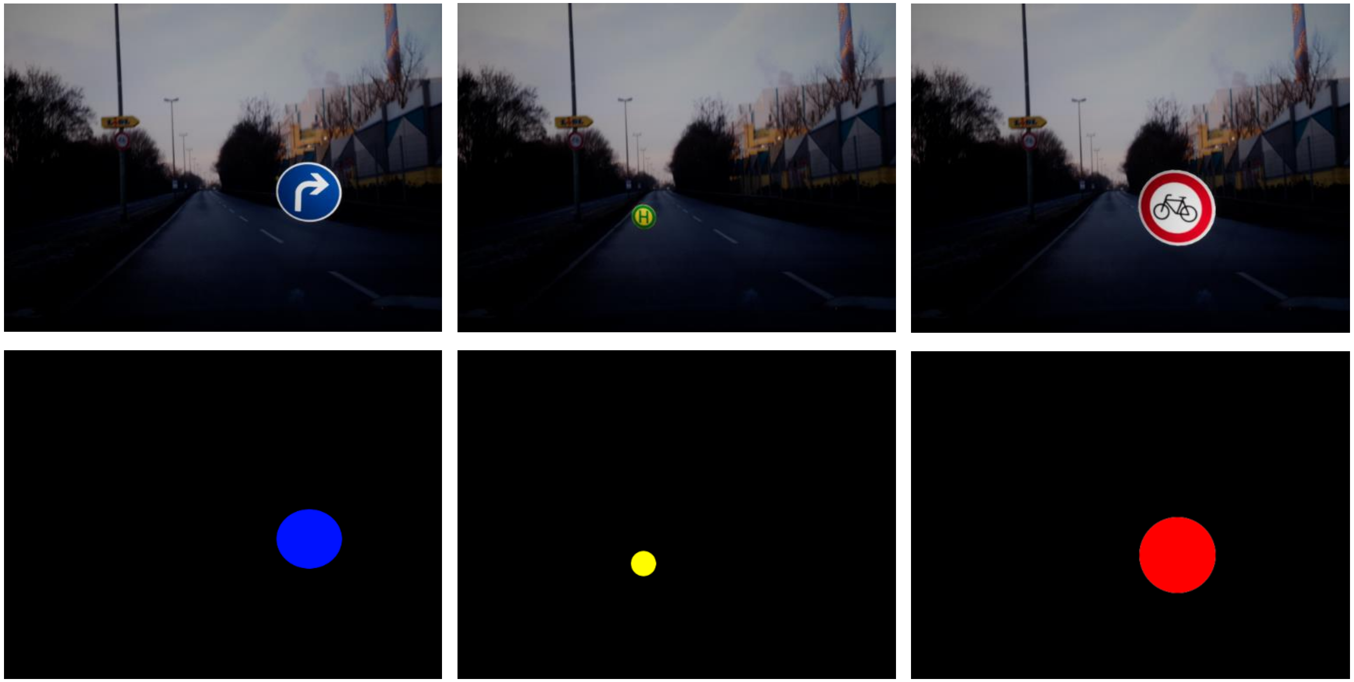
Possible source of (unlimited) situation examples – Photorealistic RGB images

Labels for each patch

Why build Simulations?

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Possible source of (unlimited) situation examples – Photorealistic RGB images



Examples
Trafficsigns

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Task

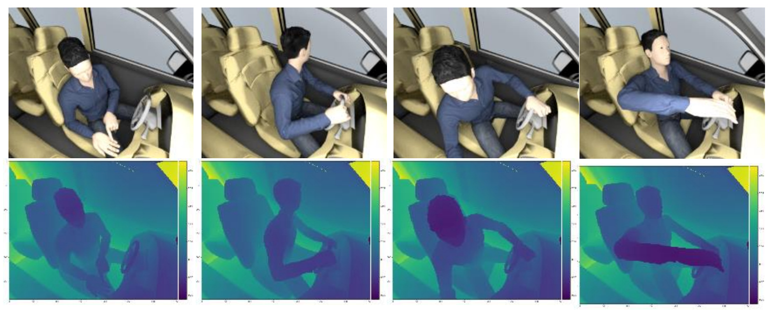
- Detect and recognize traffic signs

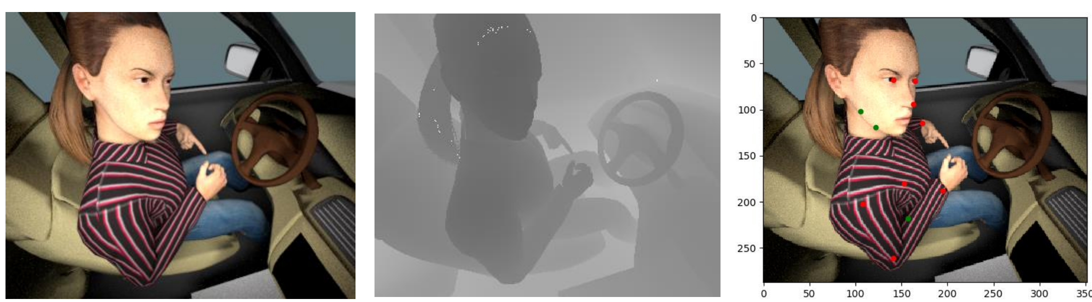
Parameters

- Θ_W
  - Geometric: CAD-desc of automotive scenes (traffic signs, streets, buildings, etc.)
  - Photometric: textures, reflectances, external lighting (context)
- Θ_P
  - Object locations (traffic-signs on poles, buildings on side of street, cars on street, etc.)
  - Governed by gravity (everything on ground-plane)
- Θ_P
  - Sensor resolution, possible exposure times, noise characteristics
  - Extrinsic (position,angle) and intrinsic (focal length, central point, distortion) parameters

Why build Simulations?

Possible source of (unlimited) situation examples – Depth data





Translation to 3D Simulation Engines

What is your scene made up of?

Geometric Parameters	Photometric Parameters	Process Parameters
Objects (Meshes) and Bounds	Material	Object poses
	Lighting	Object relations
		Temporal aspects (movements, speeds)

Is Simulation Always Helpful?

Does your simulation require very accurate complex physics? (Simulating Fluids / Simulating Infrared)

Does your scene require extensive diversity? (Facial Expressions)

Software Workflow

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

1. **Engines: RealTime VS. Raytrace**
2. **Scene Content**
3. **Simulation**
    1. Setting Up The Environment
    2. Coding
    3. Rendering
    4. Capturing Segmentation

Engines: RealTime VS. Raytrace

Raytrace

Highest photorealism

High render time

RealTime

Moderate photorealism

Rendering up to 120 frames per second

Engines: RealTime VS. Raytrace

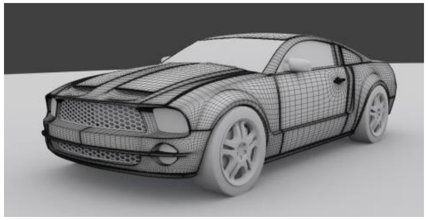
Raytrace	RealTime
CINEMA 4D	BLENDER
AUTODESK® MAYA®	CRYENGINE®
NVIDIA. OPTIX™	UNREAL ENGINE
	unity
	FROSTBITE™

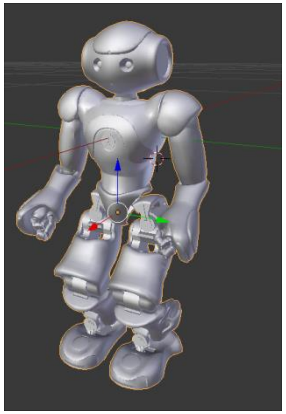
Scene Contents

1. **Virtual Environment**
   1. Models, Material, Animation
   2. Lighting
   3. Process for model placement
   4. Actor behaviour
2. **Observer (Camera)**
   1. Camera Model
   2. (Post Processing)

Real Venue

Simulated Venue





Models

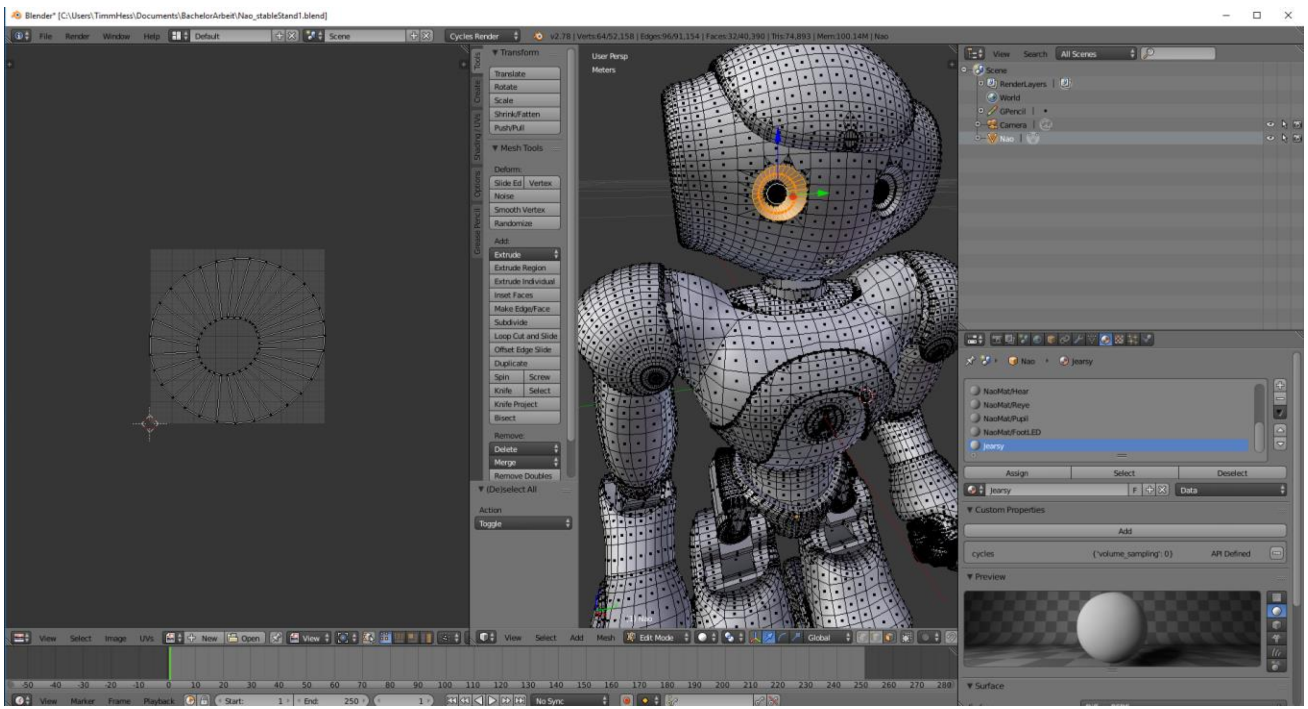
Modelling

- Designing 3D Meshes
- Making texture placement available on the mesh (UV-Unwrap)
- Animation

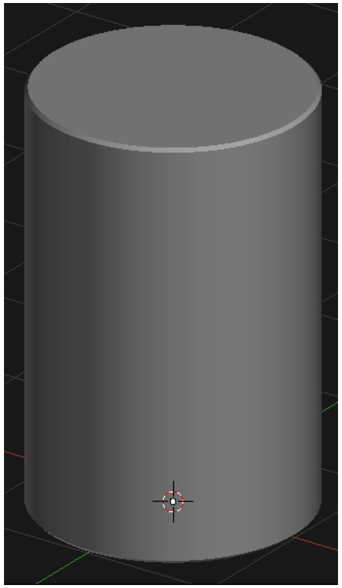
Alternative source: Online Repositories (BlendSwap, Blendermarket, …)

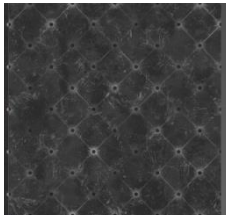
Applying Texture

UV Unwrap

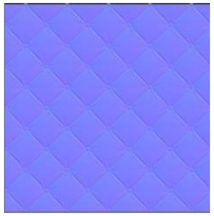


Texture / Material

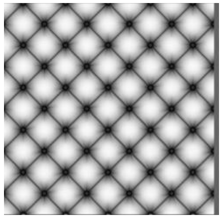




Albedo

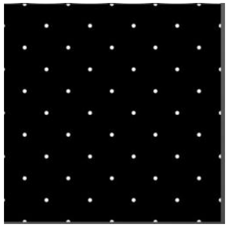


Normal



Height

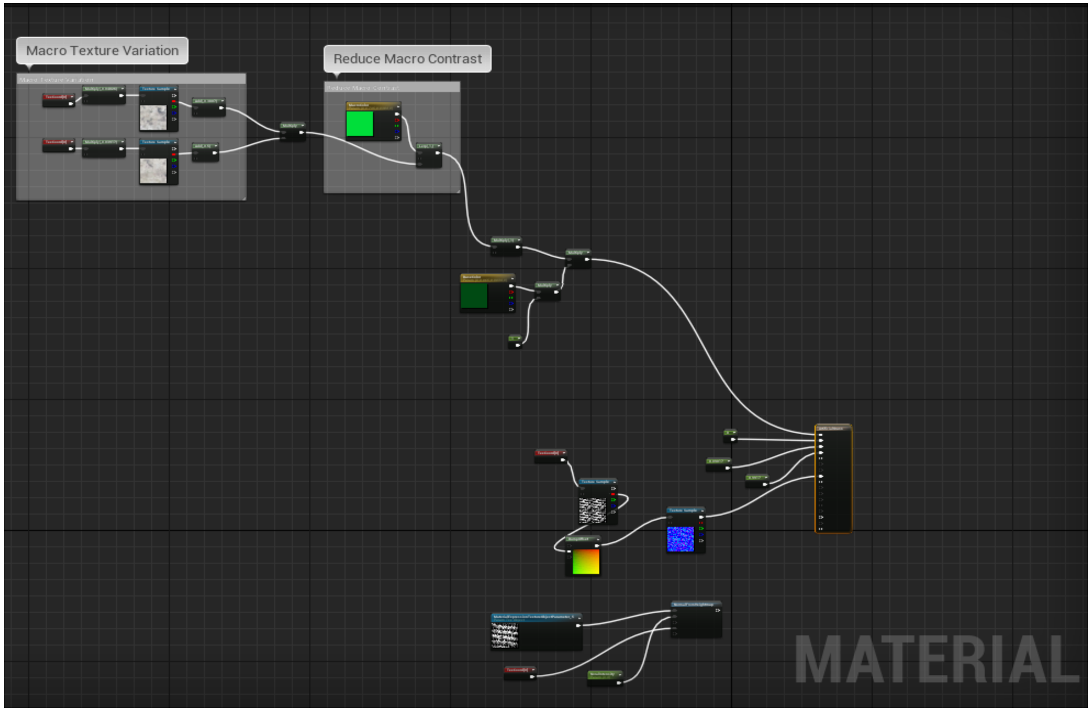
Roughness

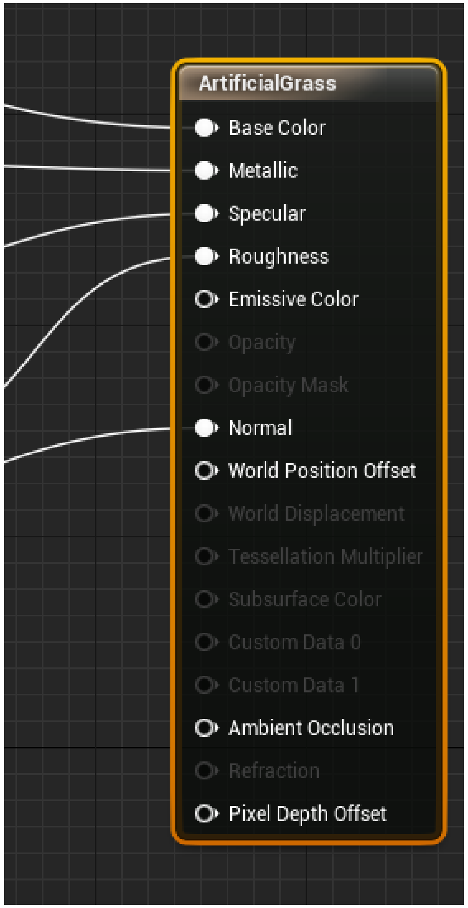


Metallic

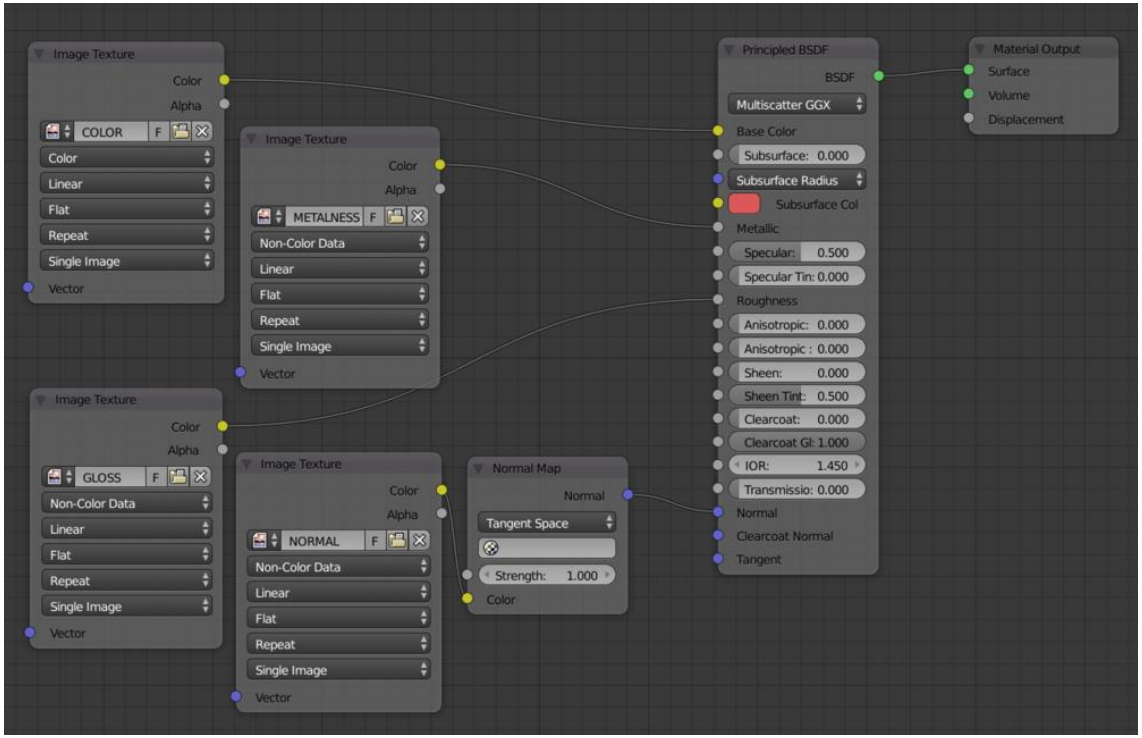
SUBSTANCE

Material (PBR)
UE4





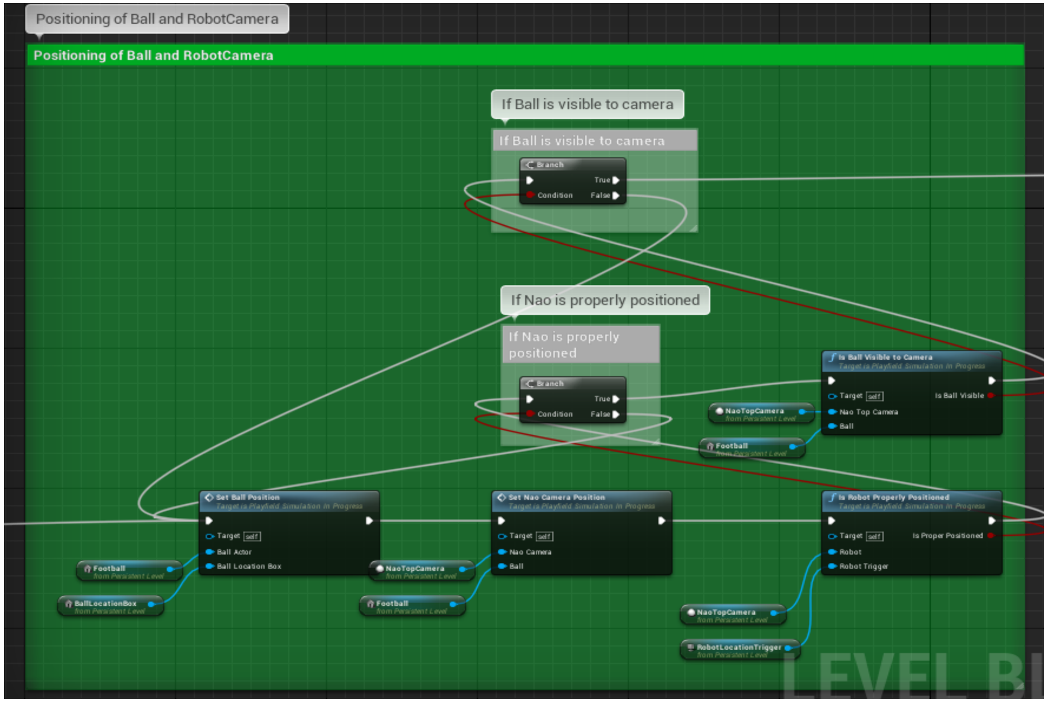
Material (PBR)
Blender



Simulation
Coding

Set software mechanisms and procedures

- Object Placement (Stochastic Scene Generation)
- Lighting Changes
- Script Activations (Rendering, Segmentation)



Simulation

Rendering

Pixel Buffer Access

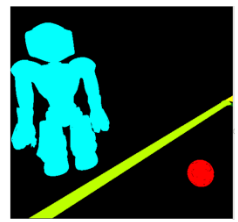
- Inherit UECameraComponent to gain access to pixel buffers
- Non trivial due to blackbox-ish rendering pipeline
- Fast

Screenshots

- Screenshot process is asynchronous! Need to freeze frames
- Slow

Simulation

Capture Segmentation



Shader

- Propagating render pipeline properties to a shader to access it
  (Multi Condition: Depth, Optical Flow, Objects)

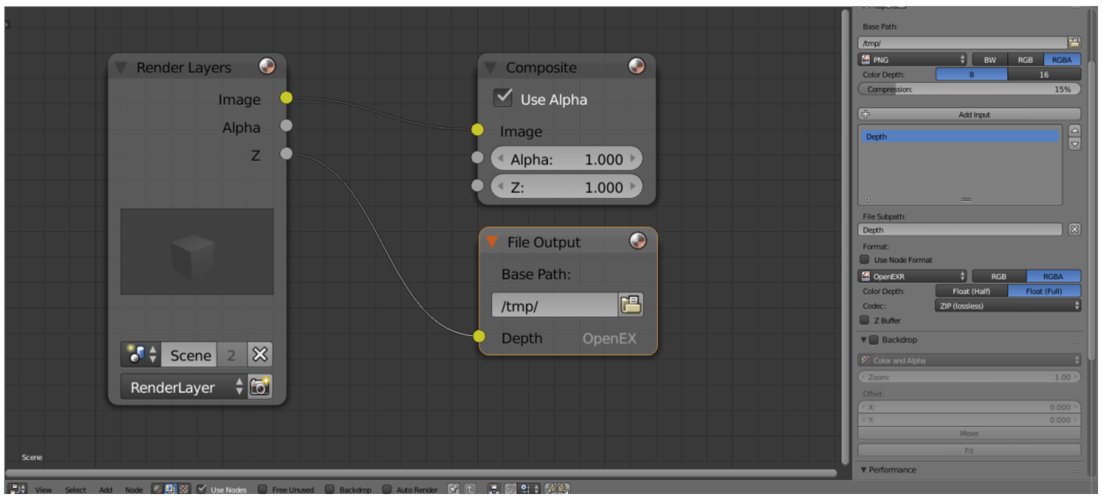
Raytracing

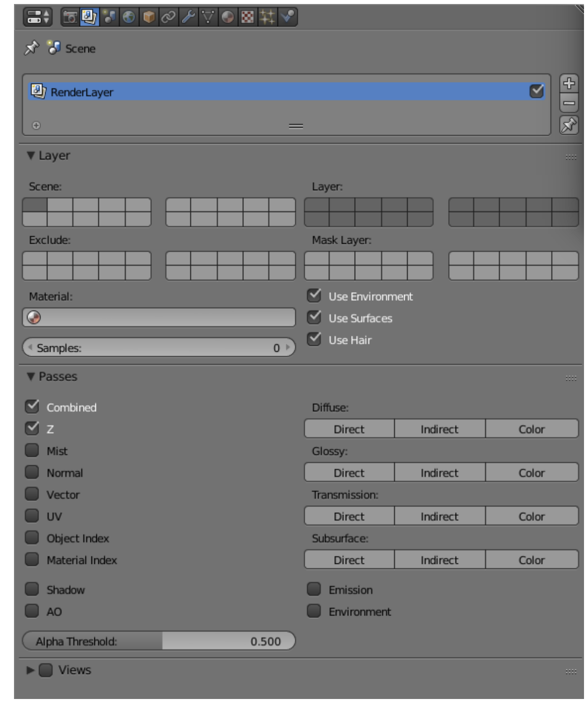
- Sending rays through all pixels in the camera viewport and listen to the first object hit by the ray
  (Slow and objects only)

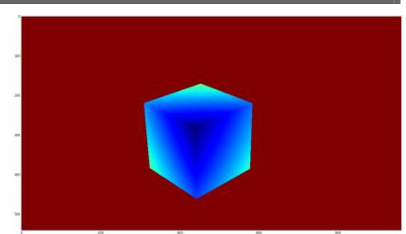
Post Processing Material

- Exploiting a engine build-in post process shader highlighting previously tagged objects
  (Fast, but error prone due to engine internals such as Antialising! Objects only)

Capture Segmentation
Renderpass - Blender

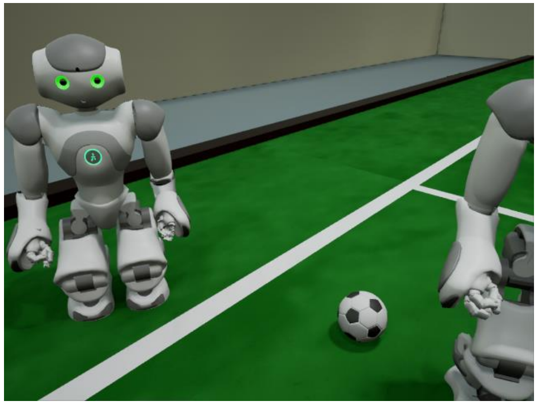


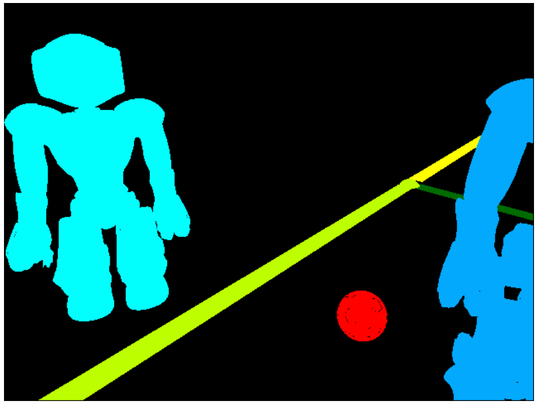




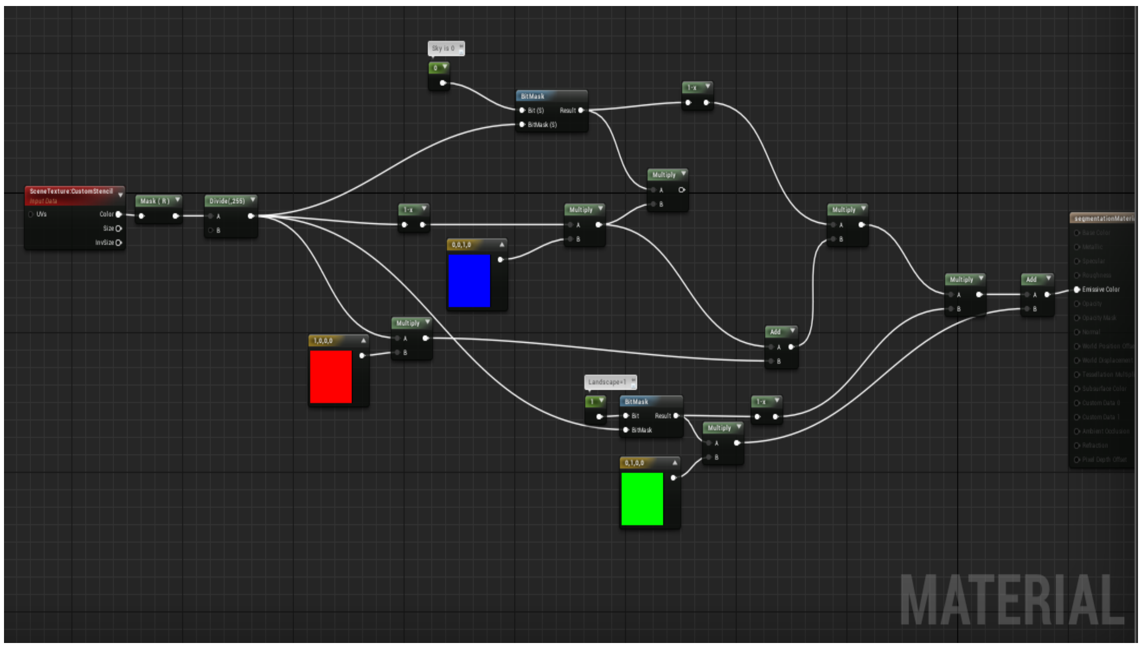
Capture Segmentation
Raytrace – UE4

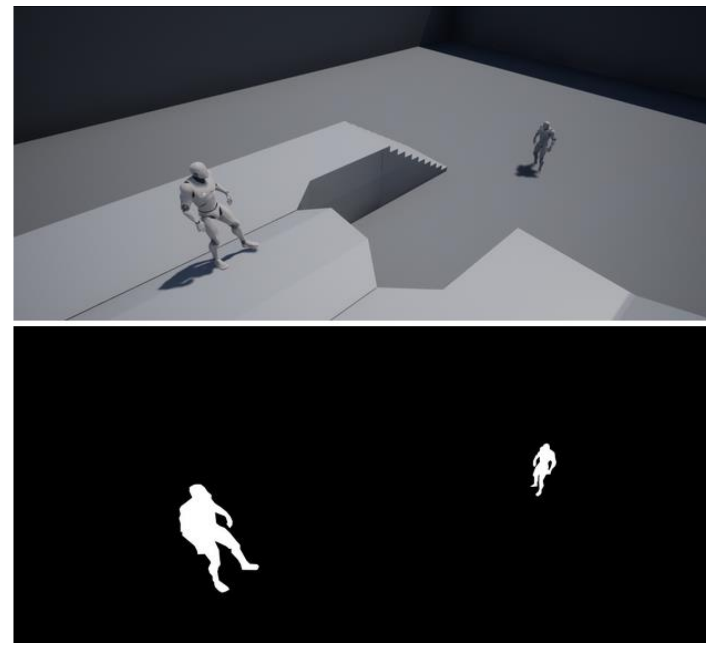
```
for (int y = 0; y < sizeY; y += stride) {
    for (int x = 0; x < sizeX; x += stride) {
        FVector2D ScreenPosition(x, y);
        FVector WorldOrigin, WorldDirection;

        DeprojectScreenToWorld(Player, ScreenPosition, WorldOrigin, WorldDirection);


        // Cast ray from pixel
        bool bHit = World->LineTraceSingleByChannel(HitResult, WorldOrigin, WorldOrigin + WorldDirection * HitResultTraceDistance, TraceChannel, CollisionQueryParams);

        AActor* Actor = NULL;
        if (bHit) {
            Actor = HitResult.GetActor();
            if (Actor != NULL) {
                bool found = false;
                for (int32 i = 0; i < nObjects; i++) {
                    if (objects[i] == Actor) {
                        FString IntAsString = FString::FromInt(i + 1);
                        outputStringMask += IntAsString + " ";
                        counter++;
                        found = true;
                        break;
                    }
                }
                if (!found) {
                    outputStringMask += "0 ";
                }
            }
            else {
                outputStringMask += "0 ";
            }
        }
        else {
            outputStringMask += "0 ";
        }
    }
    outputStringMask = outputStringMask + "\n";
```





Capture Segmentation
Post Process Shader– UE4





Existing simulation frameworks

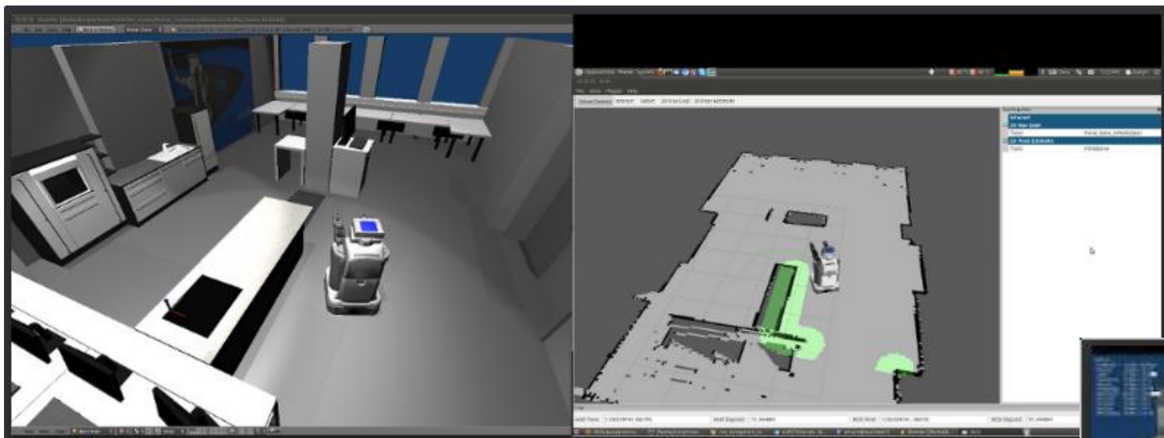
Usually provide means to

- User-control objects (cars, drones, etc.)
- Control environment, weather, illumination
- Extract most common modalities
  - Rendered images
  - Depth-maps
  - Semantic segmentation
- Some provide ``game-AI“ (e.g. cars, NPCs)

OpenSource: Microsoft - AirSim

OpenSource: CARLA (2017)

OpenSource: Morse Robot Sim



GTA – Playing for data

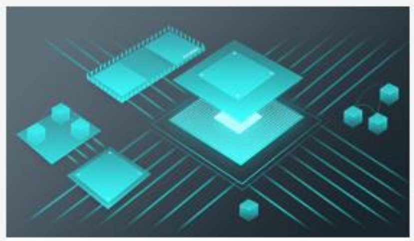
NVIDIA (2018)

Zoox (2017/18)

Baidu – Apollo Simulator

Scenarios

The simulation platform allows users to input different road types, obstacles, driving plans, and traffic light states.

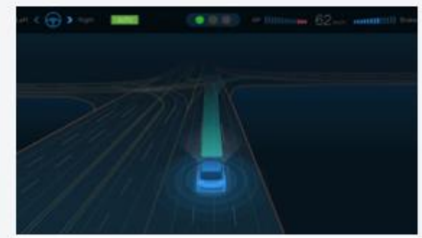


Execution Modes

Gives users a complete setup to run multiple scenarios, and upload and verify modules in the Apollo environment.

Automatic Grading System

The current Automatic Grading System tests via ten metrics: Collision detection, Traffic light recognition and logic, Speed limit, Detection of objects out of lane, End-of-route logic etc.



3D Visualization

Illustrates real-time road conditions and visualizes module output, while showing the status of the autonomous vehicle.

dSpace

And many many others

Just to list a few more:

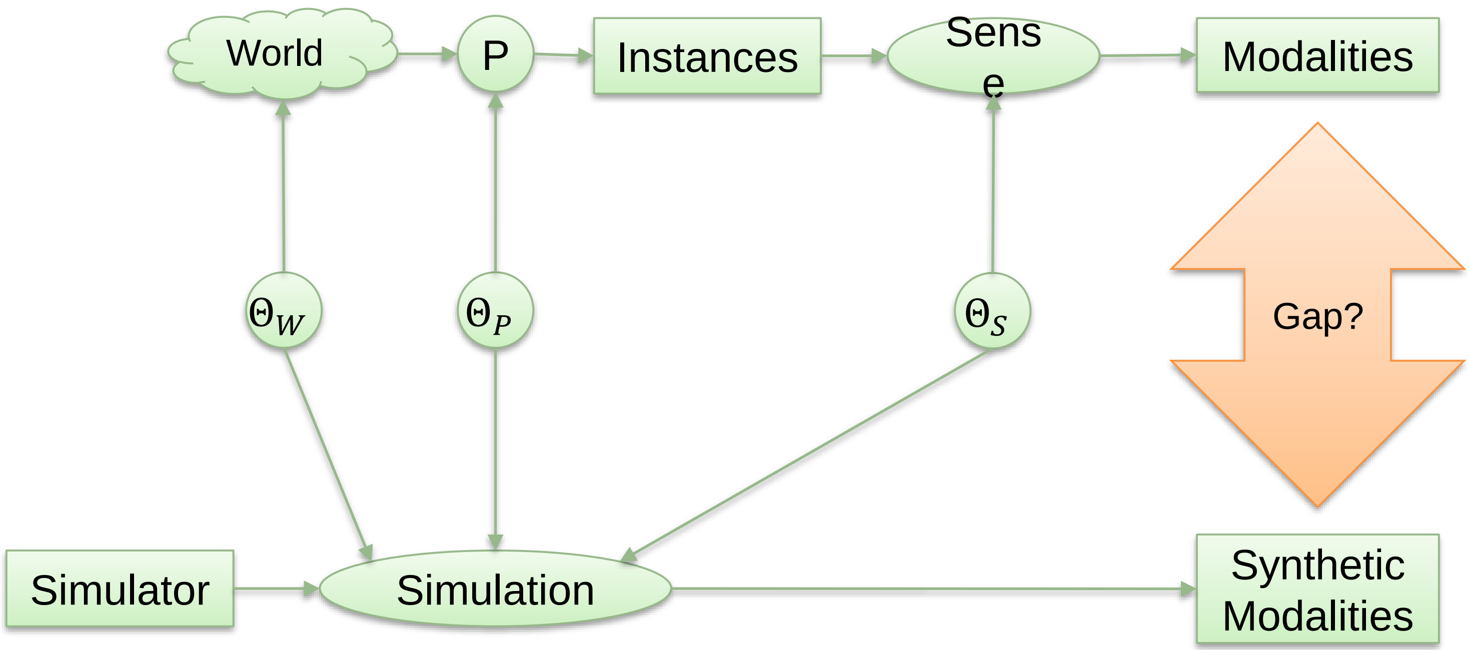
Uber, Waimo, Daimler, Siemens (Tass), AutonoVi-Sim, VIRES, rFpro, Cogna, SynCity, Simulated datasets: Virtual KITTI, Synthia

Evaluating Simulations

Does your simulation fit your case?

- Matching of color spaces
- Matching of corner case conditions
  (i.e. extreme lighting)
- How much of your problem space is covered by the simulation?

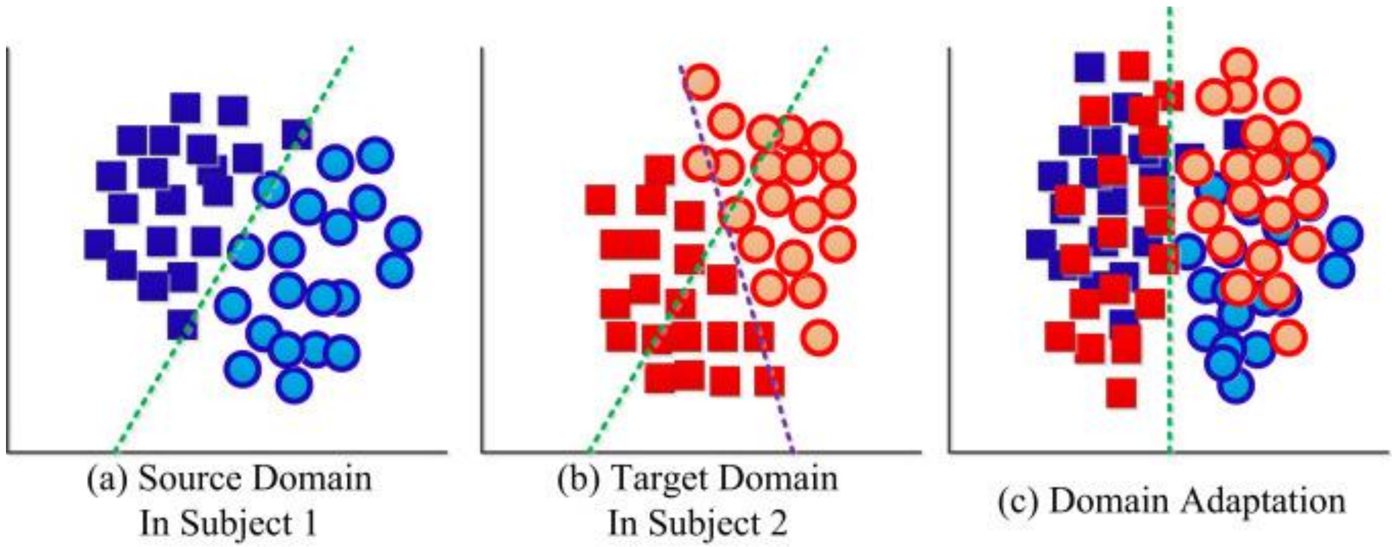
Evaluating Simulations



Training on Simulated Data
Advantages / Disadvantages

Pro	Con
Source of (unlimited) labeled data without human annotation efford **Full control over the environment**	**Obvious shifts in environment from simulated to real and subsequently mismatches in learned statistics when applied to the real world**

Domain Adaptation
Combating Domain Shift



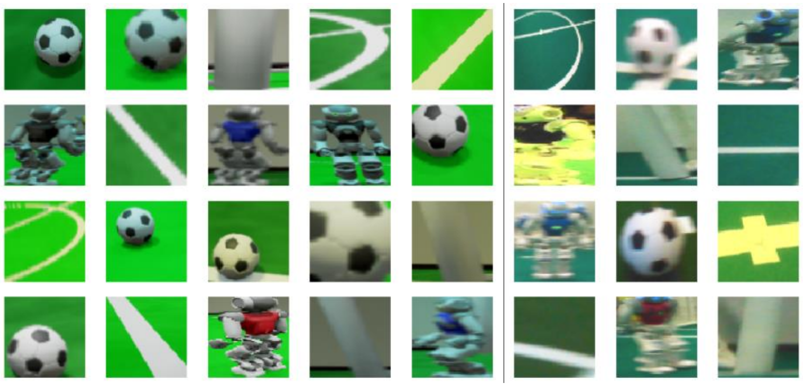
Domain Adaptation
“Rendering Fidelity”

Playing for Data: https://download.visinf.tu-darmstadt.de/data/from_games/

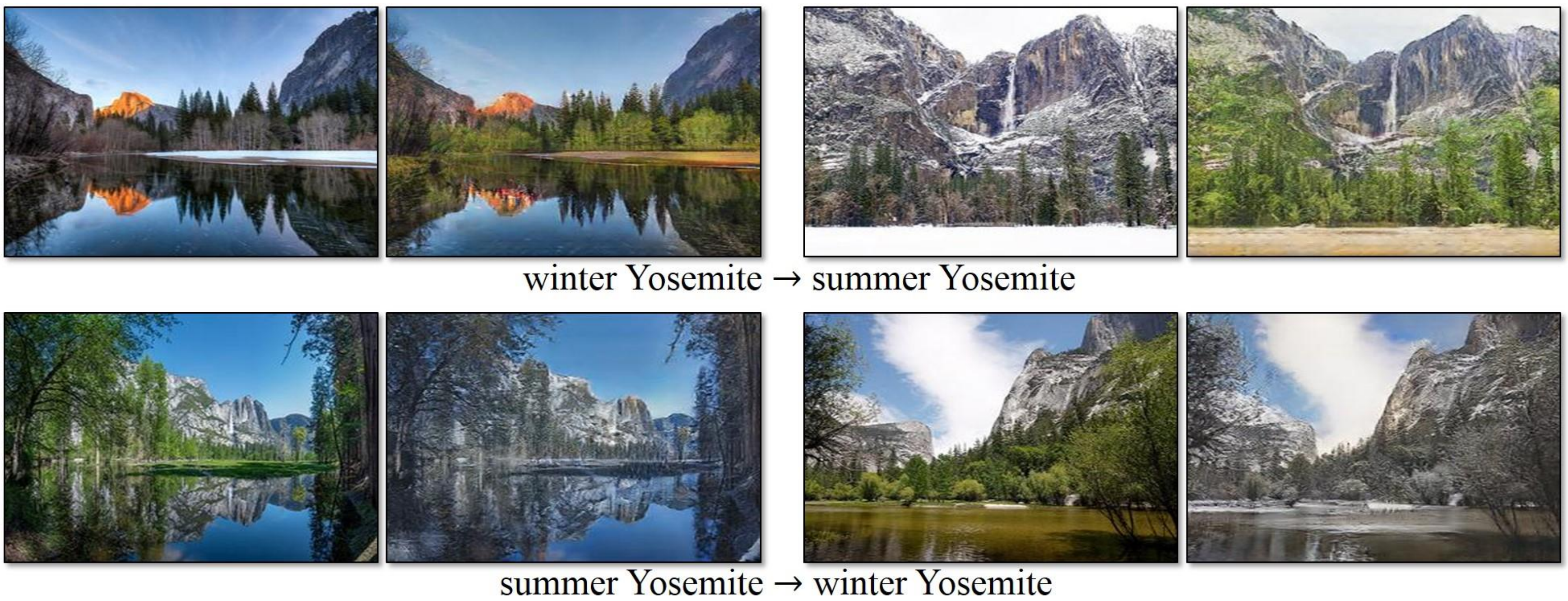
Fine Tuning

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Use additional (possibly human) labeled data from the given target domain to fine tune your model towards target space conditions



Domain Adaptation
GAN – Generative Adversarial Network



Domain Adaptation
Other Methods:

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Adversarial Methods

- Exploiting Local Feature Patterns for Unsupervised Domain Adaptation [AAAI2019]
- Domain Confusion with Self Ensembling for Unsupervised Adaptation [arXiv 10 Oct 2018]
- Improving Adversarial Discriminative Domain Adaptation [arXiv 10 Sep 2018]
- M-ADDA: Unsupervised Domain Adaptation with Deep Metric Learning [arXiv 6 Jul 2018] [Pytorch(official)]
- Augmented Cyclic Adversarial Learning for Domain Adaptation [arXiv 1 Jul 2018]
- Factorized Adversarial Networks for Unsupervised Domain Adaptation [arXiv 4 Jun 2018]
- DiDA: Disentangled Synthesis for Domain Adaptation [arXiv 21 May 2018]
- Unsupervised Domain Adaptation with Adversarial Residual Transform Networks [arXiv 25 Apr 2018]
- Simple Domain Adaptation with Class Prediction Uncertainty Alignment [arXiv 12 Apr 2018]
- Causal Generative Domain Adaptation Networks [arXiv 28 Jun 2018]
- Conditional Adversarial Domain Adaptation [arXiv 10 Feb 2018]
- Deep Adversarial Attention Alignment for Unsupervised Domain Adaptation: the Benefit of Target Expectation Maximization [ECCV2018]
- Learning Semantic Representations for Unsupervised Domain Adaptation [ICML2018] [TensorFlow(Official)]
- CyCADA: Cycle-Consistent Adversarial Domain Adaptation [ICML2018] [Pytorch(official)]
- From source to target and back: Symmetric Bi-Directional Adaptive GAN [CVPR2018] [Keras(Official)] [Pytorch]
- Detach and Adapt: Learning Cross-Domain Disentangled Deep Representation [CVPR2018]
- Maximum Classifier Discrepancy for Unsupervised Domain Adaptation [CVPR2018] [Pytorch(Official)]
- Domain Generalization with Adversarial Feature Learning [CVPR2018]
- Adversarial Feature Augmentation for Unsupervised Domain Adaptation [CVPR2018] [TensorFlow(Official)]
- Duplex Generative Adversarial Network for Unsupervised Domain Adaptation [CVPR2018] [Pytorch(Official)]
- Generate To Adapt: Aligning Domains using Generative Adversarial Networks [CVPR2018] [Pytorch(Official)]
- Image to Image Translation for Domain Adaptation [CVPR2018]

Computer Vision lectures (V. Ramesh)

- Vision as Inverse Graphics
  - Vision as Bayesian Estimation
  - History & Examples
    - MRF's for Image Segmentation (Geman & Geman)
    - Bayesian methods for various vision sub-tasks – detection, tracking, recognition, motion analysis, etc. (various authors)
    - Conditional Random Fields (Kumar et al)
    - Pattern Grammars for Vision (Zhu, Mumford)
  - Probabilistic Programming for Vision (Kulkarni et al)
- Modern Practice in ML for Vision
  - Deep CNN's, Variational Auto-encoders, Generative Adversarial Networks
  - Link between modern ML and Bayesian viewpoints

Thank you!

Backup