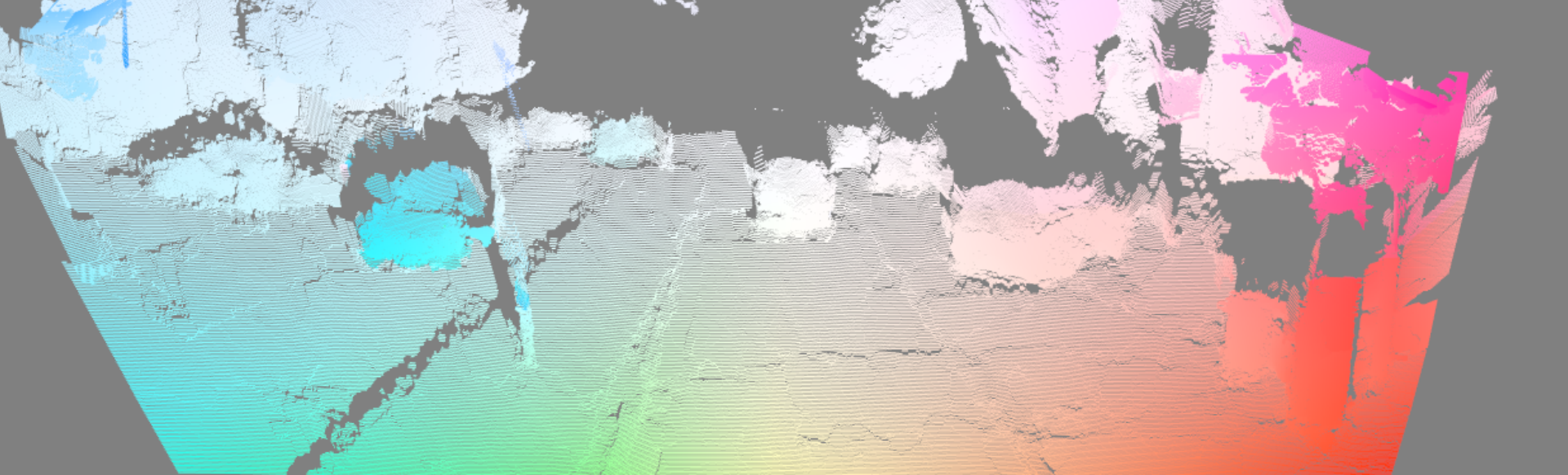
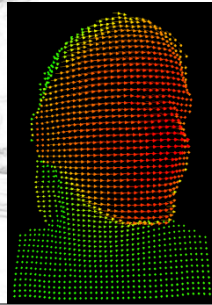
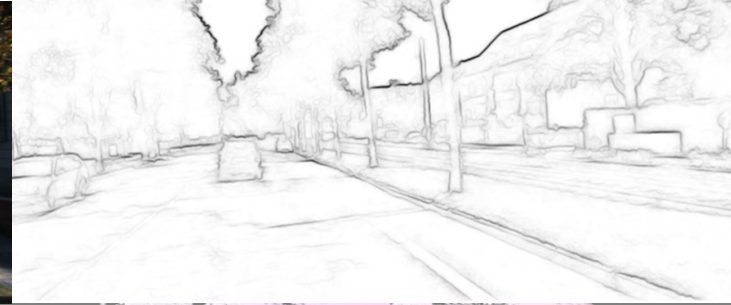




Augmented Vision - Umwelterfassung und Selbstlokalisierung



Didier Stricker

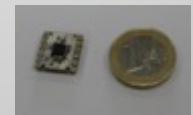
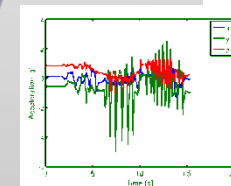
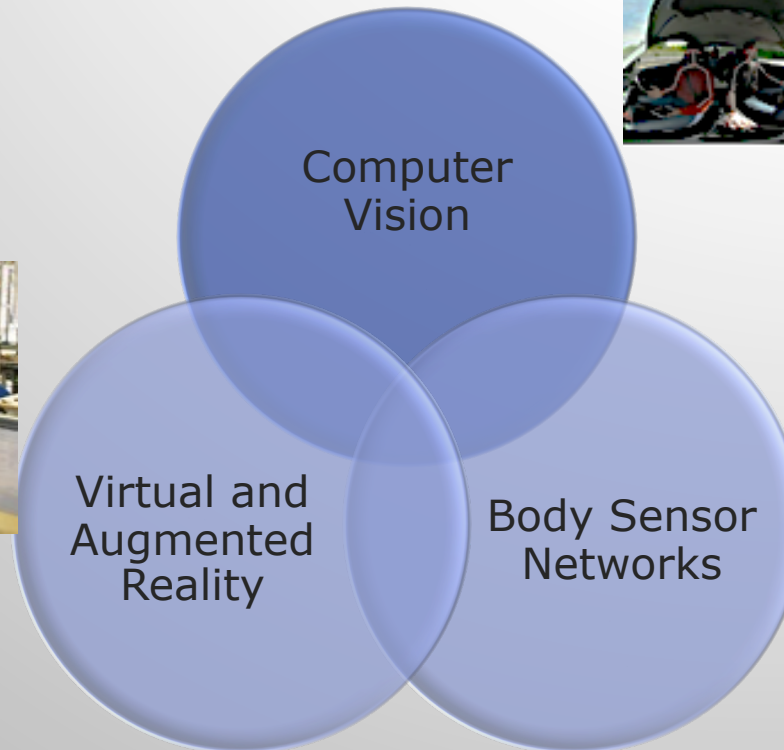
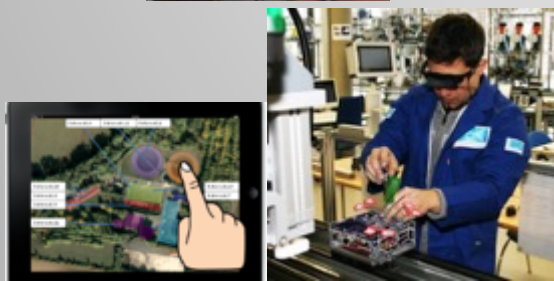
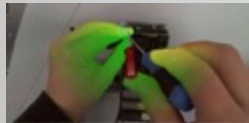
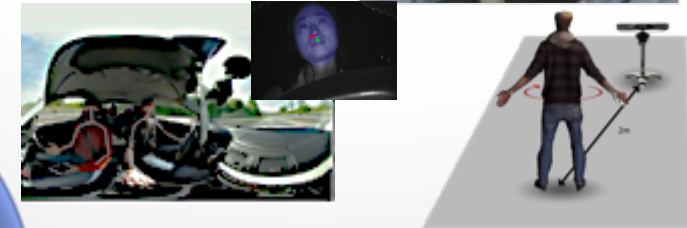
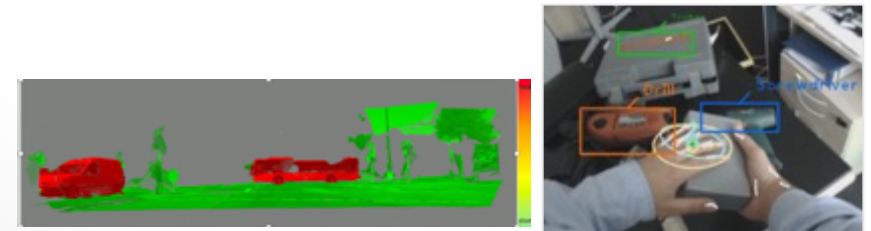


Didier.Stricker@dfki.de



Forschungsbereich Augmented Vision @ DFKI Kaiserslautern

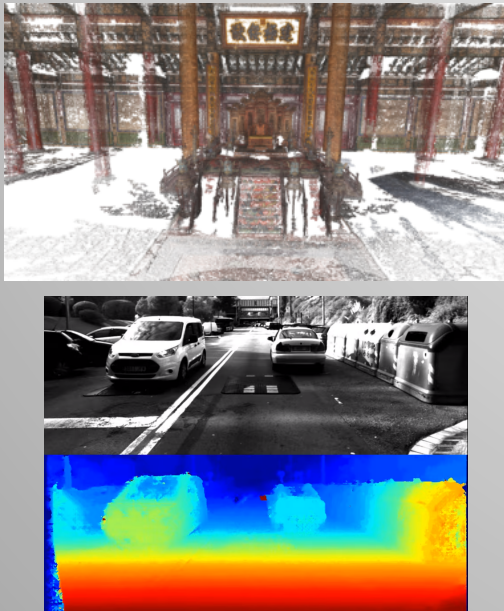
- Leitung: Prof. Didier Stricker
- 50 Doktoranden und wiss. Mitarbeiter
- Drei zusammenhängende Fachgebiete



Visuelle Wahrnehmung

Geometry

- Multiple view geometry
- Real-time stereo



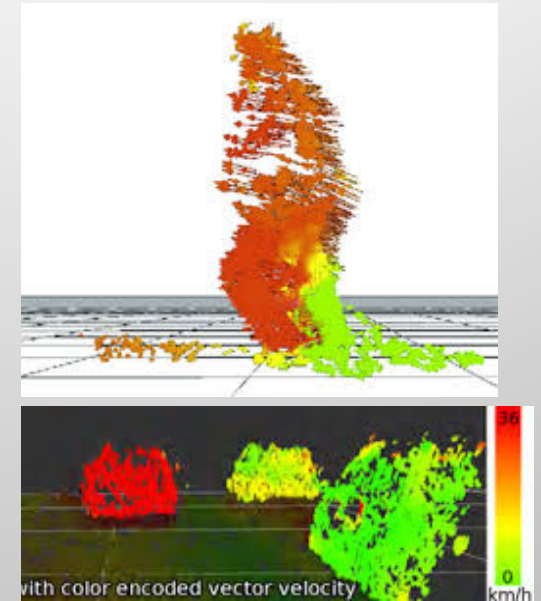
Semantic

- Object recognition
- Semantic Segmentation



Dynamics

- **Dense 4D reconstruction**
- **Temporal integration**
(in order to understand events!)



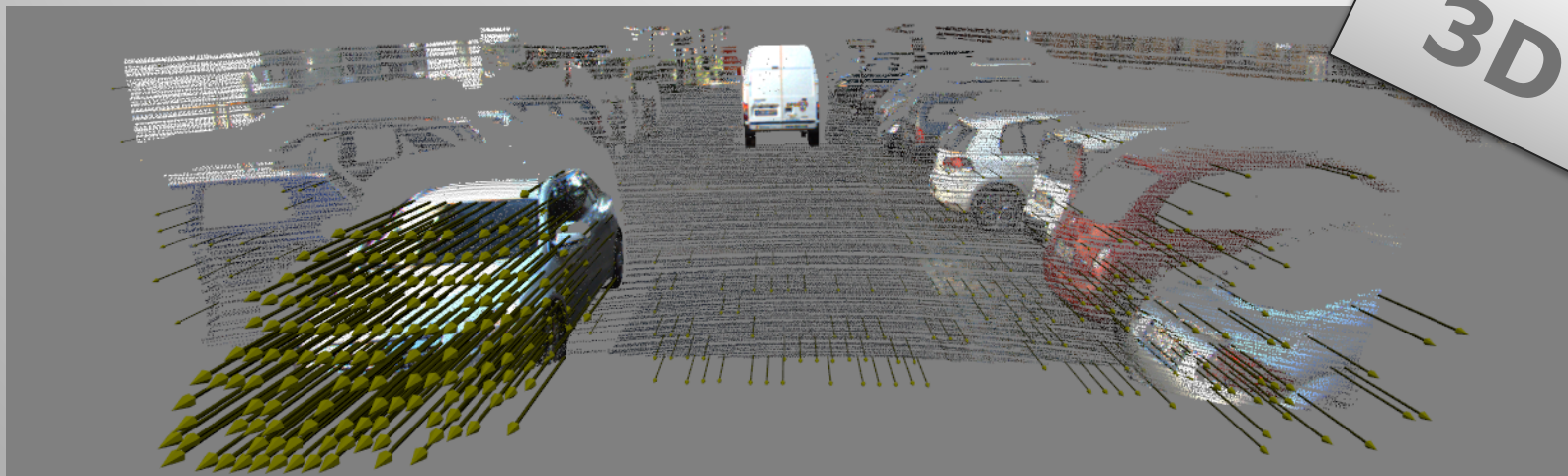
Optical Flow vs. Scene Flow

Optical Flow



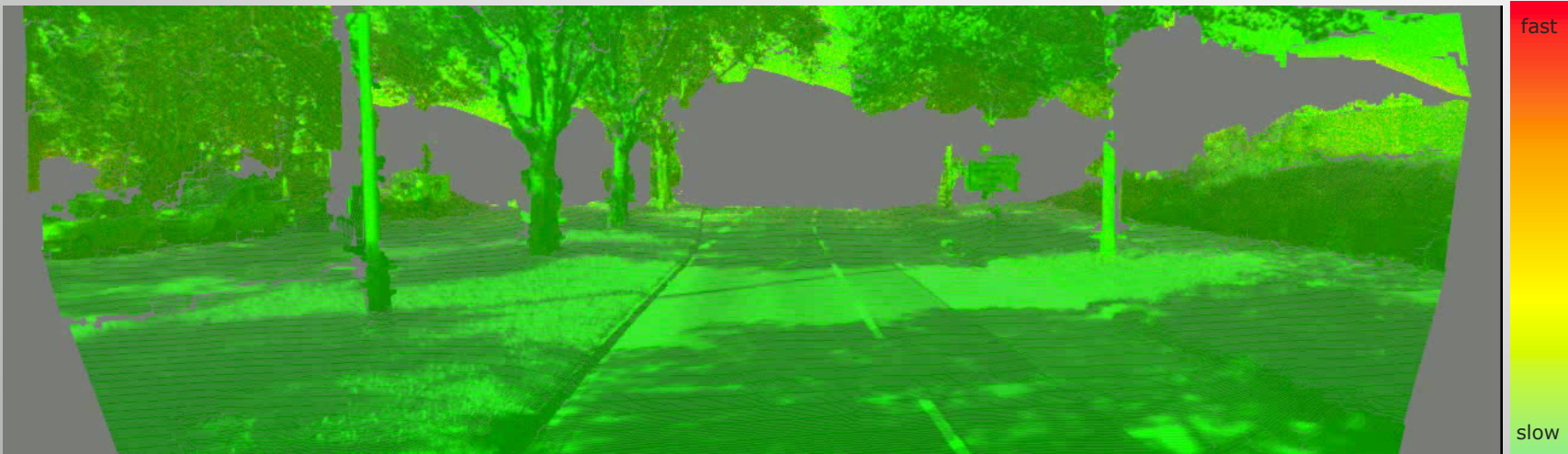
2D

Scene Flow



3D

3D Erfassung der Geometrie und Dynamik



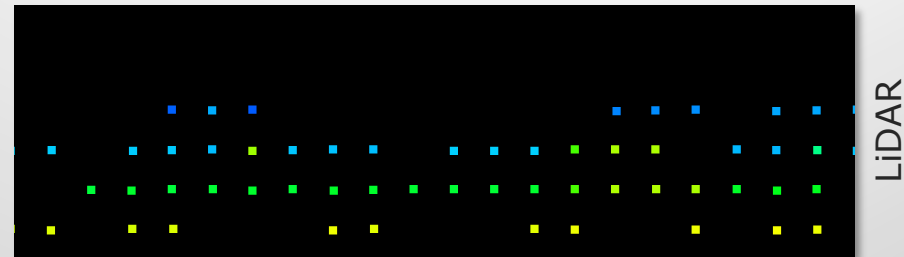
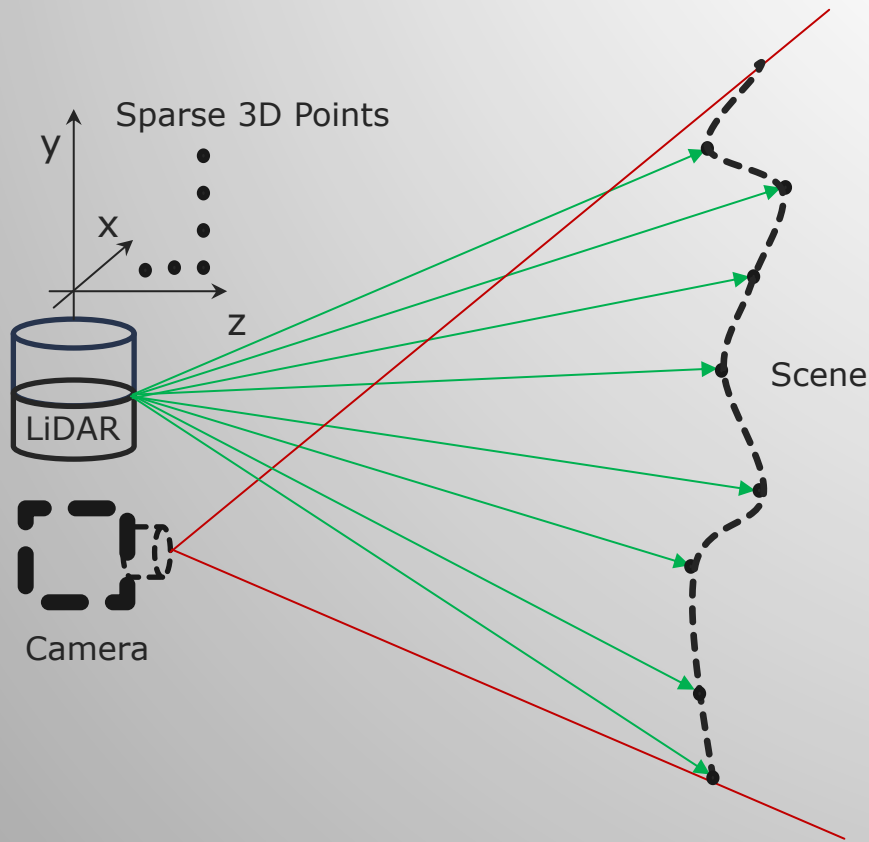
René Schuster, Oliver Wasenmüller, Georg Kusch, Christian Bailer and Didier Stricker. „Towards Flow Estimation in Automotive Scenarios“, ACM Chapters Computer Science in Cars Symposium (CSCS), 2017.

René Schuster, Christian Bailer, Oliver Wasenmüller and Didier Stricker. “Combining Stereo Disparity and Optical Flow for Basic Scene Flow”. Commercial Vehicle Technology Symposium (CVTS), Springer, 2018.

René Schuster, Oliver Wasenmüller, Georg Kusch, Christian Bailer and Didier Stricker. „SceneFlowField: Dense Interpolation of sparse scene flow correspondences“, IEEE Winter Conference on Computer Vision (WACV), 2018.

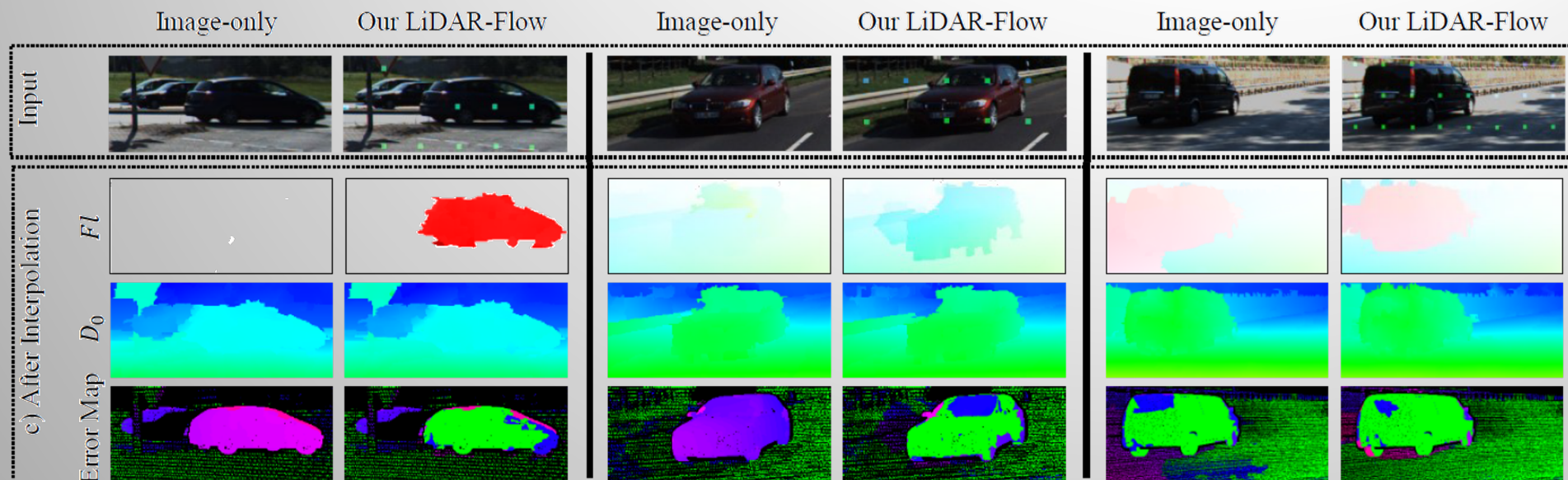
Fusion: Kamera-LiDAR

Sparse MEMS-LiDAR

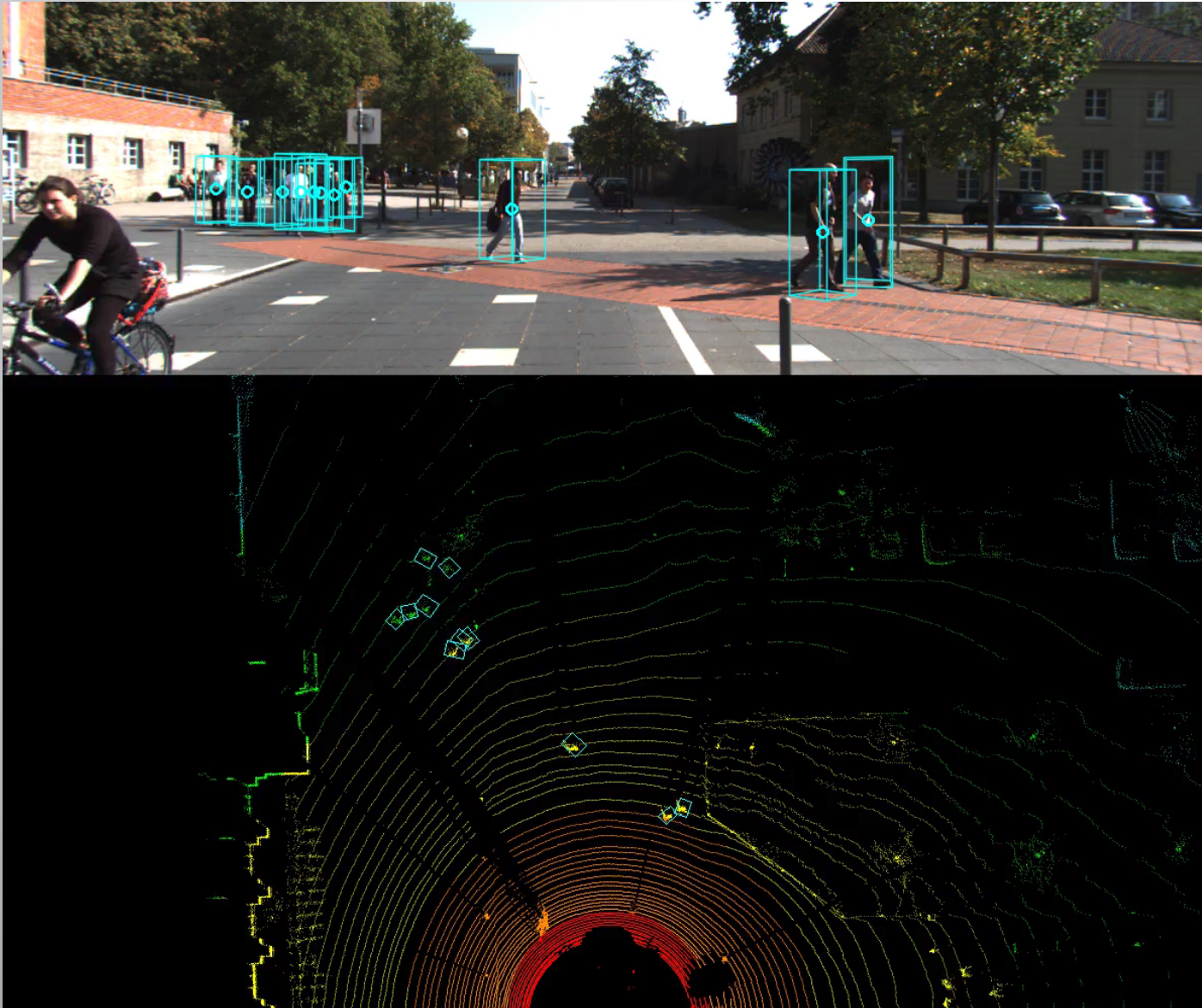


Fusion: Kamera-LiDAR

	SF Outliers [%]		
	Seeds	Window	Dense
SFF++ [4]	16.05	13.89	14.51
LiDAR-Flow	7.88	8.70	11.52
Average no. of evaluated points	88	6820	91875



Problem: 3D Fußgänger Detektion



Ergebnis: Regen



Ergebnis: Nachts



Odometrie: Lokalisierung mit LIDAR (ausführliche Evaluierung, KITTI Daten)

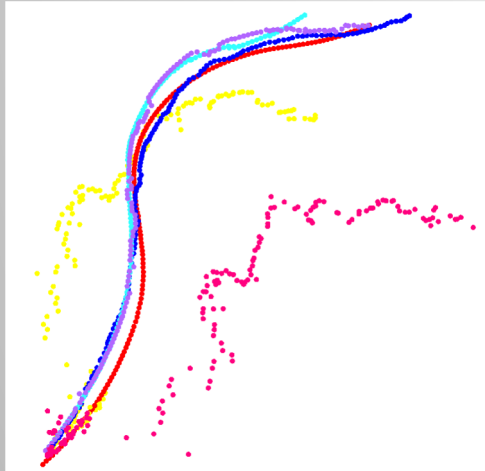
— Ground Truth Path

— FGA
(ours)

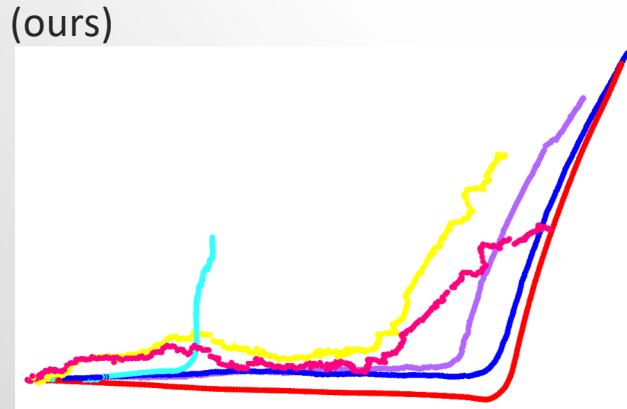
— ICP

— GMMReg

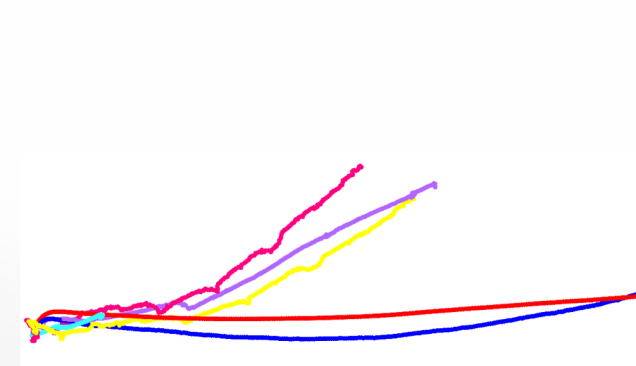
— FilterReg — CPD



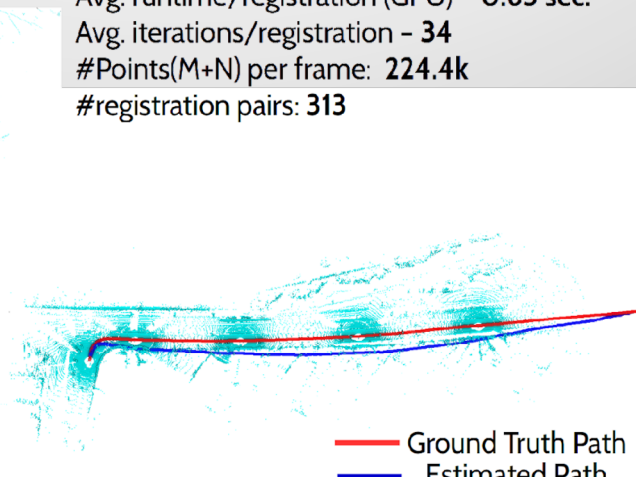
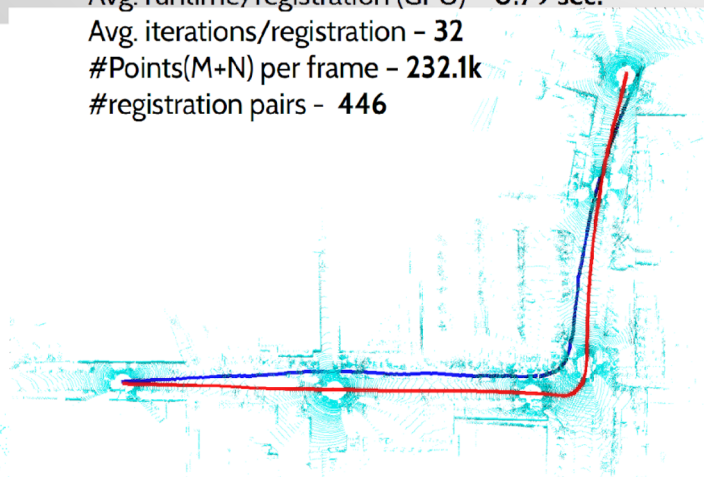
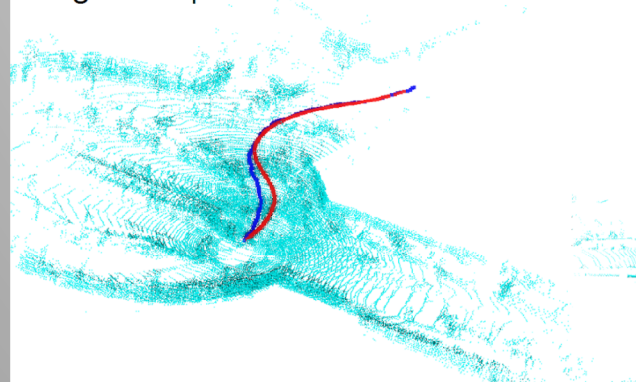
Avg. runtime/registration (CPU) – 75.8 sec.
Avg. runtime/registration (GPU) – 0.81 sec.
Avg. iterations/registration – 36
#Points(M+N) per frame – 234.2k
#registration pairs: 153



Avg. runtime/registration (CPU) – 70.3 sec
Avg. runtime/registration (GPU) – 0.79 sec.
Avg. iterations/registration – 32
#Points(M+N) per frame – 232.1k
#registration pairs – 446



Avg. runtime/registration (CPU) – 67.28 sec
Avg. runtime/registration (GPU) – 0.65 sec.
Avg. iterations/registration – 34
#Points(M+N) per frame: 224.4k
#registration pairs: 313



— Ground Truth Path
— Estimated Path

(a). 2011_09_26_drive_0005 Sequence

(b). 2011_09_26_drive_0009 Sequence

(c). 2011_09_26_drive_0014 Sequence

Human Pose Estimation



Eingebettete KI: FPGA

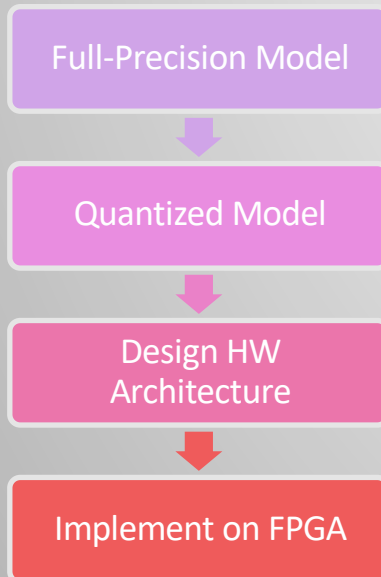
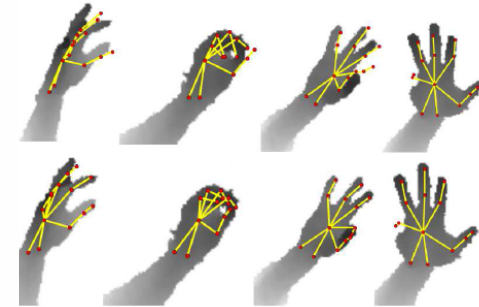


	Image Time [ms]	Effective Power [W]	Energy per Image [mJ]
FPGA	1.669	0.4	0.6676
GPU	7.01	54.98	385.41

- **4.2 Mal schneller**
- **Faktor 577 bei Energieeffizienz!**
- **Genauigkeit = ca. -2%**

DeepGUI : DragnDrop Deep Learning



Generate PyTorch Code Generate Vivado HLS Code

Save Load

```
import torch
import torch.nn as nn
class GENERATED_MODEL(nn.Module):
    def __init__(self):
        super(GENERATED_MODEL, self).__init__()
        pad2d_1 = nn.ZeroPad2d((2, 2, 2, 2))
        conv2d_1 = nn.Conv2d(1, 8, (5, 5))
        relu_1 = nn.ReLU()
        maxpool2d_1 = nn.MaxPool2d((2, 2))
        conv2d_2 = nn.Conv2d(8, 8, (5, 5))
        relu_2 = nn.ReLU()
        maxpool2d_2 = nn.MaxPool2d((2, 2))
        linear_1 = nn.Linear(6728, 1024)
        relu_3 = nn.ReLU()
        linear_2 = nn.Linear(1024, 93)
    def forward(x):
        out = self.pad2d_1(x)
        out = self.conv2d_1(out)
        out = self.relu_1(out)
        out = self.maxpool2d_1(out)
        out = self.conv2d_2(out)
        out = self.relu_2(out)
        out = self.maxpool2d_2(out)
        out = out.view(out.size(0), -1)
        out = self.linear_1(out)
        out = self.relu_3(out)
        out = self.linear_2(out)
```

PyTorch (GPU)

Generate PyTorch Code Generate Vivado HLS Code

Save Load

```
#pragma HLS dataflow

static hls::stream<float> input_stream("input stream");
static hls::stream<pad2d_1_dtype> pad2d_1_stream("pad2d_1 stream");
static hls::stream<conv2d_1_dtype> conv2d_1_stream("conv2d_1 stream");
static hls::stream<relu_1_dtype> relu_1_stream("relu_1 stream");
static hls::stream<maxpool2d_1_dtype> maxpool2d_1_stream("maxpool2d_1 stream");
static hls::stream<conv2d_2_dtype> conv2d_2_stream("conv2d_2 stream");
static hls::stream<relu_2_dtype> relu_2_stream("relu_2 stream");
static hls::stream<maxpool2d_2_dtype> maxpool2d_2_stream("maxpool2d_2 stream");
static hls::stream<linear_1_dtype> linear_1_stream("linear_1 stream");
static hls::stream<relu_3_dtype> relu_3_stream("relu_3 stream");
static hls::stream<linear_2_dtype> linear_2_stream("linear_2 stream");

#pragma HLS stream depth=1 variable=input_stream
#pragma HLS stream depth=1 variable=pad2d_1_stream
#pragma HLS stream depth=1 variable=conv2d_1_stream
#pragma HLS stream depth=1 variable=relu_1_stream
#pragma HLS stream depth=1 variable=maxpool2d_1_stream
#pragma HLS stream depth=1 variable=conv2d_2_stream
#pragma HLS stream depth=1 variable=relu_2_stream
#pragma HLS stream depth=1 variable=maxpool2d_2_stream
#pragma HLS stream depth=1 variable=linear_1_stream
#pragma HLS stream depth=1 variable=relu_3_stream
#pragma HLS stream depth=1 variable=linear_2_stream

MemoryMappedStream<float> INPUT_STREAM;
padding_multi_channel<pad2d_1_dtype, 0, 124, 124, 2, 2, 2, 2>(pad2d_1_stream, pad2d_1_stream);
conv_hwy_to_narrowconv2d_1_dtype, conv2d_1_dtype, conv2d_1_kernel_dtype, conv2d_1_bias_dtype, 1, 8, 128, 128, 5, 5>(conv2d_1_stream, conv2d_1_kernel, conv2d_1_bias, conv2d_1_stream);
relu_crelu_1_dtype, 131072>(relu_1_stream, relu_1_stream);
pool_multi_channel<maxpool2d_1_dtype, 0, 124, 124, 2, 2>(maxpool2d_1_stream, maxpool2d_1_stream);
conv_hwy_to_narrowconv2d_2_dtype, conv2d_2_dtype, conv2d_2_kernel_dtype, conv2d_2_bias_dtype, 8, 8, 62, 62, 5, 5>(conv2d_2_stream, conv2d_2_kernel, conv2d_2_bias, conv2d_2_stream);
relu_crelu_2_dtype, 3072>(relu_2_stream, relu_2_stream);
pool_multi_channel<maxpool2d_2_dtype, 0, 64, 64, 2, 2>(maxpool2d_2_stream, maxpool2d_2_stream);
fc_linear_1_dtype, linear_1_dtype, linear_1_w_dtype, linear_1_b_dtype, 6728, 1024>(linear_1_stream, linear_1_w, linear_1_b, linear_1_stream);
relu_crelu_1_dtype, 1024>(relu_3_stream, relu_3_stream);
fc_linear_2_dtype, linear_2_dtype, linear_2_w_dtype, linear_2_b_dtype, 1024, 93>(linear_2_stream, linear_2_w, linear_2_b, linear_2_stream);
```

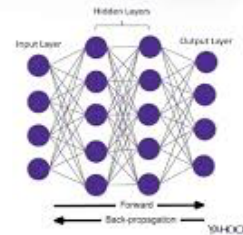
HLS C++ (FPGA)

Ausblick

- Fusion von **Modalitäten** und **Methoden**
- Eingebettete Lösung oder „**Edge AI**“ wird eine zentrale Rolle spielen
- „**Continual Learning**“ - Lernen mit wenigen Beispielen und mit reduzierter Hardware
- Von PKW zu Arbeitsmaschinen: **Autonomes Fahren und autonomes Arbeiten (Rauhe Umgebungen)**



Deep Learning



Thank you for your attention!



DFKI GmbH

Department Augmented Vision
Trippstadterstr. 122
D-67663 Kaiserslautern



Didier Stricker



Didier.stricker@dfki.de



<http://av.dfki.de/>