# Spatio-thermal depth correction of RGB-D sensors based on Gaussian Processes in real-time

Christoph Heindl

PROFACTOR GmbH, Austria

# Introduction

# RGBD - Commodity 3D Sensing

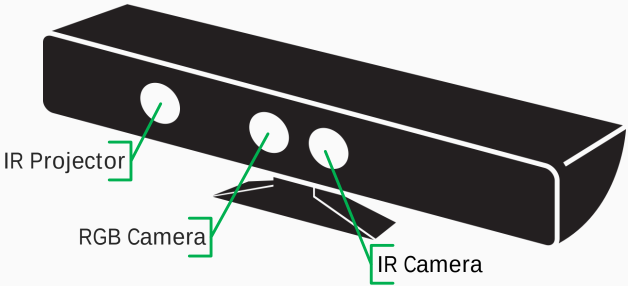



**Figure 1:** RGBD-Sensor components

Main vision sensor in many robotic / computer vision applications these days.

# RGBD - Commodity 3D Sensing

### Advantages

- Dense depth maps in real-time
- Simultaneous color and depth streams
- Affordable devices

### Drawbacks

- Inaccurate compared to industrial cameras
- Operating temperature sensitivity
- No time synchronization

# Our contribution

Improve the accuracy of dense depth maps by correcting in a joint spatio-thermal domain.

# Gaussian Processes

Gaussian Process: a probabilistic model used in supervised machine learning for classification and regression[1, 2].

Regression searches for an unknown function $\mathcal{F}$ that satisfies

$$\mathcal{F}(\mathsf{X}) = \mathsf{y} + \epsilon$$

where $\mathsf{X} \in \mathbb{R}^{N \times d}$ are training features, $\mathsf{y} \in \mathbb{R}^{N \times 1}$ are the target scalar observations and $\epsilon \sim \mathcal{N}$ is i.i.d. Gaussian observation noise.

# Gaussian Process Regression

In a probabilistic framework we would like to compute a distribution over functions based on training data and use the posterior predictive distribution for new predictions.

$$p(\mathsf{y}_\star|\mathsf{X}_\star, \mathsf{X}, \mathsf{y}) = \int p(\mathsf{y}_\star|\mathcal{F}, \mathsf{X}_\star)p(\mathcal{F}|\mathsf{X}, \mathsf{y})d\mathcal{F}$$

where $\mathsf{X}_\star$ are test inputs, $\mathsf{y}_\star$ are the predictions, $\mathsf{X}$ are training inputs, $\mathsf{y}$ are training target values.

Generally intractable to integrate over all possible functions and their parametrizations.

Assumption: all variables are jointly Gaussian

$$\begin{bmatrix} \mathsf{y} \\ \mathsf{y}_\star \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mathcal{M}(\mathsf{X}) \\ \mathcal{M}(\mathsf{X}_\star) \end{bmatrix}, \begin{bmatrix} \Sigma(\mathsf{X},\mathsf{X}) & \Sigma(\mathsf{X},\mathsf{X}_\star) \\ \Sigma(\mathsf{X},\mathsf{X}_\star)^T & \Sigma(\mathsf{X}_\star,\mathsf{X}_\star) \end{bmatrix} \right)$$

where $\mathcal{M} : \mathbb{R}^{K \times d}$ is a mean function, and $\Sigma$ is a positive-definite function encoding the covariance between samples.

# Gaussian Process Regression

Posterior predictive now simplifies due to Gaussian identities

$$
\begin{aligned}
p(\mathsf{y}_\star | \mathsf{X}_\star, \mathsf{X}, \mathsf{y}) \sim \mathcal{N}&(\mathsf{y}_\star | \mathsf{u}_\star, \mathsf{K}_\star), \\
\mathsf{u}_\star &= \mathcal{M}(\mathsf{X}_\star) + \Sigma(\mathsf{X}, \mathsf{X}_\star)^T \Sigma(\mathsf{X}, \mathsf{X})^{-1} (\mathsf{y} - \mathcal{M}(\mathsf{X})), \\
\mathsf{K}_\star &= \Sigma(\mathsf{X}_\star, \mathsf{X}_\star) - \Sigma(\mathsf{X}, \mathsf{X}_\star)^T \Sigma(\mathsf{X}, \mathsf{X})^{-1} \Sigma(\mathsf{X}, \mathsf{X}_\star)
\end{aligned}
$$

Computationally interesting: just linear algebra.

Covariance functions encode assumptions about the form of functions that are being modelled. Squared exponential kernel function

$$\Sigma_{ij} = \sigma_s^2 \exp(-0.5(\mathsf{x}_i - \mathsf{x}_j)^T \mathsf{W}(\mathsf{x}_i - \mathsf{x}_j)) + \delta_{ij}\sigma_y^2.$$

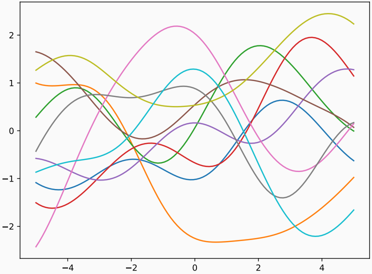
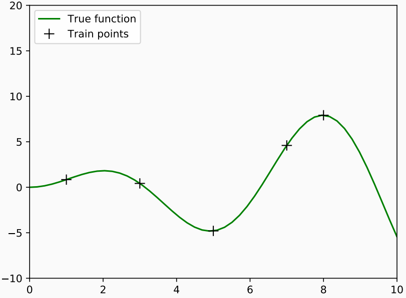
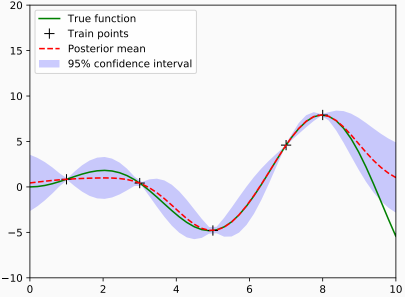



Figure 2: Ten samples from the squared exponential prior.




(a)




(b)

**Figure 3:** Illustration of Gaussian Process regression. Left: Input samples and true function, right: Predictions including confidence intervals. Output generated by [3].

# Approach and Results

# RGBD Depth Correction

Per pixel dense depth correction using Gaussian Process regression based on spatio-thermal input. For a given depth map $\mathcal{D}$ and temperature reading $t$ we compute a corrected depth map $\mathcal{D}_\star$ by

$$\mathcal{D}_\star(i,j) = \mathcal{D}(i,j) + \Delta_{ij}$$

where $\Delta_{ij}$ is given by a Gaussian Process Regression $\Delta_{ij} = \mathcal{G}(\mathsf{x}_{ij})$.

# RGBD Depth Correction

The input to the Gaussian Process $\mathcal{G}$ is given by

$$\mathsf{x}_{ij} = \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} = \begin{bmatrix} \mathcal{D}(i,j)\mathsf{K}^{-1}[i,j,1]^T \\ t \end{bmatrix}$$

$$y_{ij} = \mathcal{D}_{\mathrm{RGB}}(i,j) - \mathcal{D}_{\mathrm{IR}}(i,j)$$

# Setup

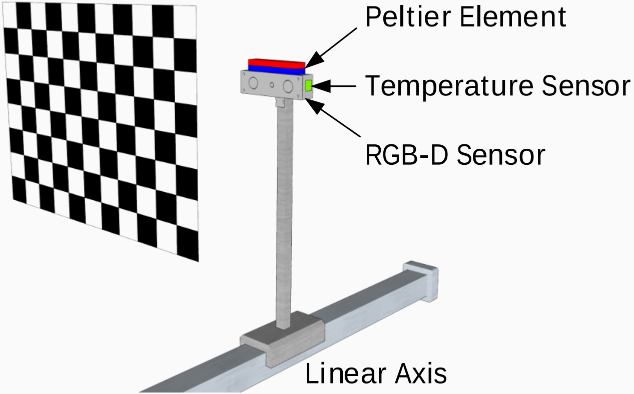



Figure 4: Hardware capture setup

# Temperature Influence

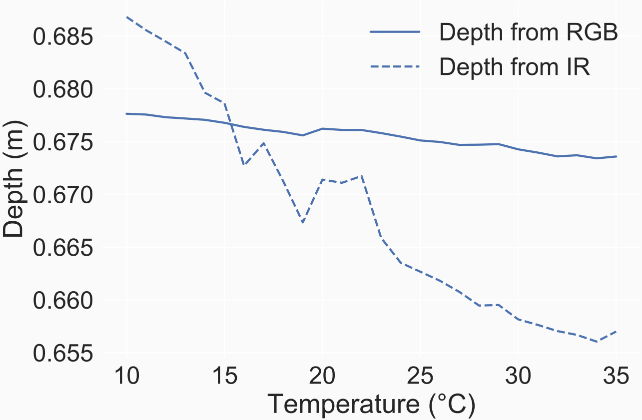



**Figure 5:** Effects of varying temperature on depth estimation at a fixed distance over a small window region.

# Results


**Figure 6:** Depth correction by Gaussian Process regression. Before/after correction effects. The white-speckles in after-images are due to missing depth sensor readings for which no correction can be computed. The error is significantly reduced independent from chosen position and temperature.

# Results


**Figure 7:** Depth correction by Gaussian Process regression.

# Results


Figure 8: Depth correction by Gaussian Process regression.

# Results

**Table 1:** RMSE before and after correction in Cartesian space across all temperature and distance captures. The correction yields an improvement by one order of magnitude.

| | $x$ (mm) | $y$ (mm) | $z$ (mm) |
|---|---|---|---|
| RMSE before correction | 5.7 | 3.7 | 16.0 |
| RMSE after correction | 0.7 | 0.5 | 2.2 |

# Performance

Gaussian Process predications can leverage GPUs. Mostly boils down to linear algebra matrix multiplications.

**Table 2:** Execution times for correction per depth map frame of size 640 times 480. The GPU optimized version outperforms the other variants.

| | Time (s) | FPS (1/s) |
|---|---|---|
| CPU | 20 | 0.05 |
| GPU optimized | 0.14 | 7.1 |

# Open Data

For reproducibility we made our dataset and source code publicly available at

`https://github.com/cheind/rgbd-correction`

# References

Carl Edward Rasmussen and Christopher KI Williams.
*Gaussian processes for machine learning*, volume 1.
MIT press Cambridge, 2006.

Kevin P Murphy.
*Machine learning: a probabilistic perspective.*
MIT press, 2012.

Christoph Heindl.
Gaussian processes in python.
`https://github.com/cheind/py-gaussian-process`, 2017.

# Acknowledgements

This work was supported by

Federal Ministry
for Transport,
Innovation and Technology