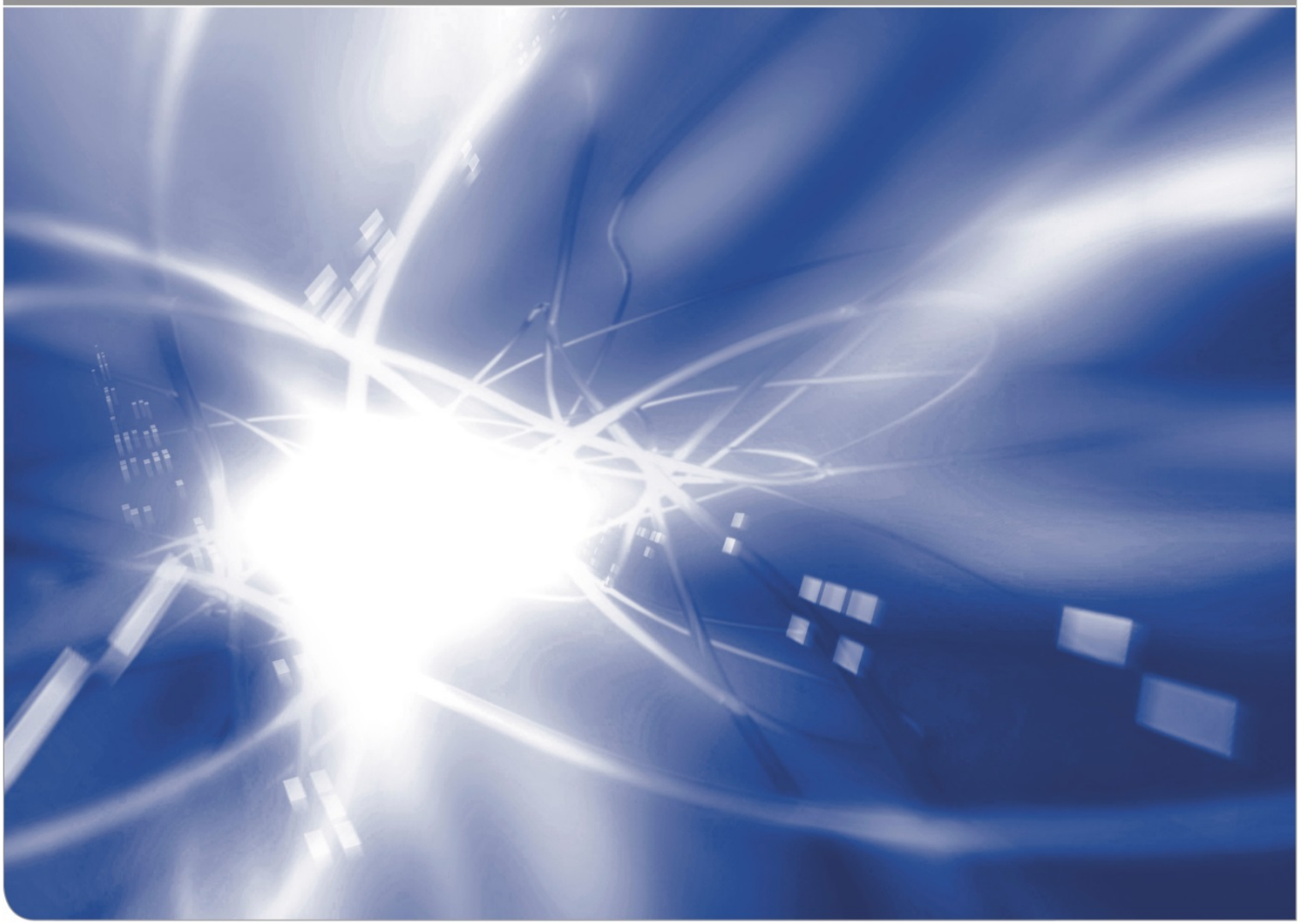
Karlsruhe Institute of Technology

Proceedings of the 2nd Workshop on Proximity Perception in Robotics at IROS 2019, Macau, China

Edited by

Hosam Alagi[1], Stephan Mühlbacher-Karrer[2], Stefan Escaida Navarro[3], Björn Hein[1,4], Hubert Zangl[5] and Keisuke Koyama[6]

November 8th, 2019, Macau, China.
IEEE/RSJ International Conference on Intelligent Robots and Systems



Introduction

In this workshop, the benefits of *Proximity Perception* in robotics, especially to those unfamiliar with it, have been highlighted. An important aspect was to encourage networking within the research community as well as to bridge the gap to the industry supporting technology transfer.

Regarding the broader objectives of the workshop, we observe that *Proximity Perception* technologies have the potential to play an essential role in service, industrial robotics, Human-Robot Collaboration (HRC), compliant robotics applications and even bio-inspired robotics. On the one hand, designs of robotic graspers that include Proximity Sensors enable novel control strategies for exploration, grasping and manipulation. On the other hand, the sensors allow safety features to fulfill leading technical specifications such as ISO/TS 15066 for the operation of collaborative robots and improve the autonomy and perception of robotic systems in all fields.

These proceedings contain the papers accepted and presented during the poster session of the workshop.

Talks held at the workshop:

Full body proximity sensing and human-robot interaction via sensor/actuator augmented skins
Nikolaus Correll,
University of Colorado at Boulder

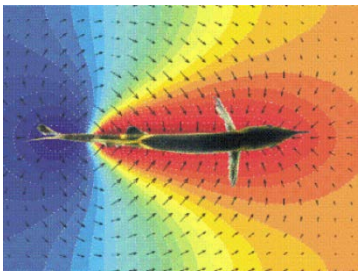
Proximity sensors for grasping applications in robotics
Jelizaveta Konstantinova,
Ocado Technology and Queen Mary University of London

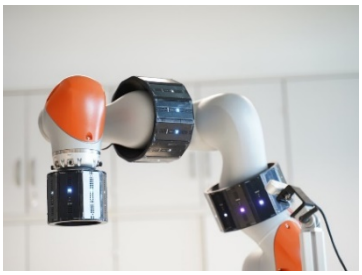
Electric field sensing for bio-inspired underwater robotics
Frédéric Boyer and Vincent Lebastard,
IMT Atlantique Bretagne-Pays de la Loire Ecole Mines-Telecom

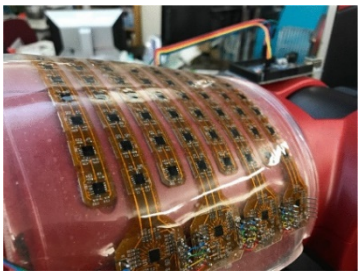
Vision-based tactile sensor FingerVision for fragile object manipulation
Akihiko Yamaguchi,
Tohoku University

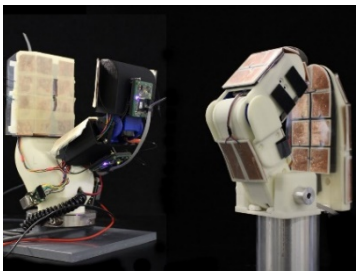
Reliable and context related grasping for autonomous mobile robots
Hansruedi Früh,
F&P Robotics

LiDAR and 3D Imaging for Robotics and Automated Mobility
Norbert Druml,
Infineon Technologies AG









For more information, please visit the workshop website.
proxelsandtaxels.org

Enhanced Human-Machine Interaction by Combining Proximity Sensing with Global Perception

Christoph Heindl*, Markus Ikeda*, Gernot Stübl*, Andreas Pichler* and Josef Scharinger**

***Abstract*— The raise of collaborative robotics has led to wide range of sensor technologies to detect human-machine interactions: at short distances, proximity sensors detect nontactile gestures virtually occlusion-free, while at medium distances, active depth sensors are frequently used to infer human intentions. We describe an optical system for large workspaces to capture human pose based on a single panoramic color camera. Despite the two-dimensional input, our system is able to predict metric 3D pose information over larger field of views than would be possible with active depth measurement cameras. We merge posture context with proximity perception to reduce occlusions and improve accuracy at long distances. We demonstrate the capabilities of our system in two use cases involving multiple humans and robots.**

I. INTRODUCTION

Proximity perception is an active research field, aiming to equip robotics with nontactile near-field sensors to advance robot autonomy and human-machine interoperability. While proximity sensing is a compelling concept on close-up range, it fails to recognize spatio-temporal events on moderate distances from the robot [1]. These events, however, provide important information to create a more robust and intuitive set of interaction patterns.

In previous works, optical systems were increasingly used to close this information gap. Many approaches integrate active depth cameras to detect human key points and measure distances between objects in the environment. The downsides of active depth sensors are the short operating range, the sensitivity to extraneous light and the increased occlusion caused by the camera-projector arrangement.

We describe an optical system that complements proximity perception by observing human and machines from a bird's-eye perspective (see Figure 1). Even though our system works solely with color images from one single panoramic camera, it delivers pose information for humans and robots in a metric space. We combine the provided pose information with near-field measurements from proximity sensors to robustly detect human-machine interaction patterns. We demonstrate the benefits of merging complementary input modalities in two use cases involving multiple robots and humans [1].

*Visual Computing and Robotics, PROFACTOR GmbH, Austria christoph.heindl@profactor.at

**Institute of Computational Perception, Johannes Kepler University, Austria josef.scharinger@jku.at

[1]Demonstration video https://youtu.be/1X0xF3m36BA

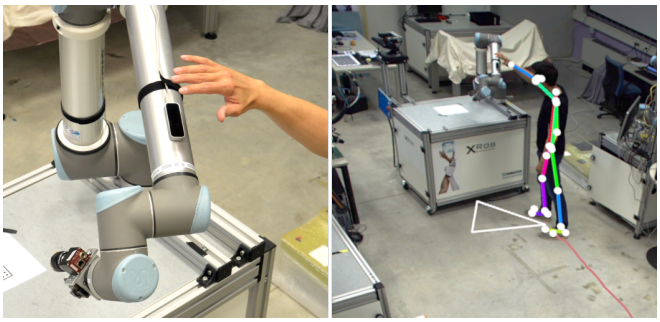


Fig. 1. Our system complements proximity sensing (left) with global perception from a bird's-eye perspective in real-time (right). Based on a color input image, we estimate the following features shown superimposed in the right image: human pose (dots connected by lines), worker orientation (white triangle) and the recent movement trajectory (red path on floor). The majority of these measurements are available in a metric 3D space and combined with proximity perception to coordinate human-machine interactions.

A. Related Work

Vision based human-machine interaction has been studied before. Guanglong et al. [2] combine RGB-D cameras and inertial measurement units to detect human gestures for robot learning. Zimmermann et al. [3] use depth-based devices to record human instructions for teaching a service robot.

Systems based on depth cameras are also used to study human-robot safety aspects. Fabrizio et al. [4] describe a depth sensing enabled device to compute distances to dynamic obstacles for collision avoidance. Švarnỳ et al. [5] propose using 2D keypoint detection merged with RGB-D data to measure distances between a robot and a single human operator.

B. Contributions

The proposed system has a number of benefits. Our system works with a readily available single wide-angle color camera, eliminating the range limitations of active depth devices. The inference step builds on recent advances in pose estimation, enabling our system to answer complex image related queries robustly. In addition to instantaneous pose, we track individual humans to provide pose trajectories over time. Although the detection step takes place in image space, the majority of predictions are elevated to a metric 3D space, enabling natural fusion with near-field sensors to create a richer set of human-machine interactions.

2nd Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China

II. METHOD

Figure 3 illustrates the core components of our system. These are detailed in the following sub-sections.

A. View Synthesis

In order to process panoramic color images, we first synthesize one or more synthetic rectilinear views (see Figure 2). View synthesis assumes a virtual pinhole camera P that is potentially rotated with respect to the physical source frame S. Next, every virtual camera pixel $\mathbf{u}^P$ is mapped to a corresponding source pixel $\mathbf{u}^S$ using the method described in [6]. The computed pixel position is bilinearly interpolated to determine the final color value associated with $\mathbf{u}^P$.

Fig. 2. View synthesis from panoramic color images. A virtual rectilinear camera image (right) is created from a highly distorted spherical image region (left).

B. Pose Estimation

To compute the 2D pose of humans and robots, we first predict 2D keypoints from synthesized color views (see Figure 7 b,d). We use neural network architectures, depicted in Figure 4 based on works of Cao et al. [7] and Heindl et al. [8] to perform keypoint localization. Each network, composed of a series of Deep Convolutional Neural Networks (DCNNs), regresses keypoint coordinates by transforming the input color images $\mathbf{x}^P \in \mathbb{R}^{3\times H\times W}$ into keypoint belief maps $\hat{\mathbf{b}}^P \in \mathbb{R}^{C\times H\times W}$. Here H, W are image height and width and C is the number of output keypoints (6 for robots, 25 for humans). Each belief map encodes the likelihood of observing a particular keypoint in a specific image region. We train these networks with both real [7] and artificially generated images [8].

In a subsequent step, 2D poses are transformed into a metric space via a set of homographies. In particular, we propose the use of an image-to-ground homography $\mathbf{H}_P^G \in \mathbb{R}^{3\times 3}$, to map image positions to metric ground coordinates as follows

$$\mathbf{u}^G = \mathrm{D}\left(\mathbf{H}_P^G \left[u_x^P, u_y^P, 1\right]^T\right), \tag{1}$$

where $\mathrm{D}(\cdot)$ is the dehomogenization operator $\left[x', y'\right] = \left[x/z, y/z\right]$. Because such mappings are accurate only for body parts sufficiently close to the ground (such as foot positions), we use a statistical body model to gain the ability to map extra keypoints such as hips and shoulders.

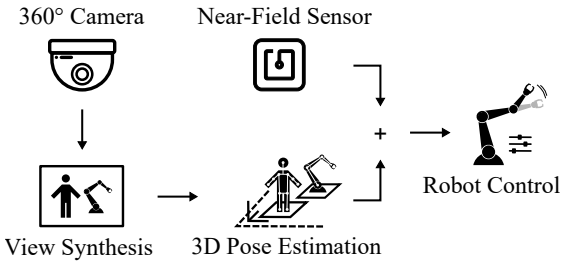


Fig. 3. System Overview. Our approach considers panoramic images as input; these are synthesized to form one or more rectilinear views. Deeply learned neural networks then predict 2D human and robot pose keypoints. These detections are subsequently lifted to a 3D metric space using homographies of planes. Next, our system fuses near-field measurements with human/robot pose context to create location-aware events. These events in turn lead to environmental reactions defined by the application scenario.

These additional points serve to predict body orientation and to stabilize body positions in case of partial occlusions. Regarding robots, we map only the base joints to determine their location.

C. Multiple Object Tracking

We filter and track human trajectories using a Kalman filter, assuming a linear dynamic motion model (see Figure 1, 7d). All operations are performed in ground plane coordinates to avoid perspective effects. To assign newly detected poses to existing ones, we create a bipartite graph: Given a set of pose detections $\mathcal{D}^t$ and a set of forward predicted poses $\mathcal{P}^t$ at time t, we add an edge e_{dp} for every possible combination of detected $d \in \mathcal{D}^t$ and tracked $p \in P^t$ poses to the graph. The cost C_{dp} associated with edge e_{dp} is computed as the Euclidean distance between the ground positions of d and p. We use the method of Kuhn et al. [9] to solve for the optimal assignments and update the Kalman filters correspondingly.

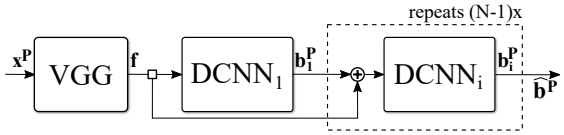


Fig. 4. Joint localization architecture [8]. First, base features $\mathbf{f}$ are extracted from input image $\mathbf{x}^P$. An initial Deep Convolutional Neural Network (DCNN) performs initial joint belief prediction $\mathbf{b}_1^P$. A series of DCNNs then refines belief prediction to generate the final prediction $\hat{\mathbf{b}}^P$.

D. Merging Pose Information with Proximity Measurements

Pose information alone has already proven useful in our experiments with human-robot interaction. However, line-of-sight occlusions and large object distances limit the system's ability to measure small movements accurately. Therefore, we merge global pose context with gestures detected by a proximity device[2].

[2] https://www.leapmotion.com/

In particular, we treat data fusion as a probabilistic classification problem and consider the presence of gestures intended for robot r to be latent random variables $C \in \{0, 1\}$. At each time-step we observe the following noisy features: a) a confidence level of the (possibly carried) proximity sensor for an active gesture $F_g \in [0, 1]$, b) the relative position of the operator closest to the robot in ground plane coordinates $F_{xy} \in \mathbb{R}^2$ and c) the orientation of the operator with respect to the robot, expressed as the cosine of the angle $F_o \in [-1, 1]$. Given the observations $\mathcal{F} = \{F_g, F_{xy}, F_o\}$, we compute the posterior probability of the presence of a command $\mathrm{p}(C \mid \mathcal{F})$ using Bayes theorem

$$\mathrm{p}(C \mid \mathcal{F}) = \frac{\mathrm{p}(\mathcal{F} \mid C)\mathrm{p}(C)}{\mathrm{p}(\mathcal{F})}.$$

For computational reasons, we introduce the following independence assumptions

$$(f \perp\!\!\!\perp g \mid C) \quad \forall f, g \in \mathcal{F}, f \neq g.$$

That is, all observed features are independent from each other given the state of command C. Thus, the posterior probability simplifies to

$$\mathrm{p}(C \mid \mathcal{F}) = \frac{1}{Z}\,\mathrm{p}(C) \prod_{f \in \mathcal{F}} \mathrm{p}(f \mid C),$$

where Z is the partition function given by

$$Z = \sum_{c \in \{0,1\}} \mathrm{p}(C{=}c) \prod_{f \in \mathcal{F}} \mathrm{p}(f \mid C{=}c).$$

Our model, shown graphically in Figure 5, follows the structure of a naïve Bayes classifier. We assume the following underlying probability distributions

$$\begin{aligned} C &\sim \text{Bernoulli}(\theta) \\ F_g \mid C{=}c &\sim \text{Beta}(a_c, b_c) \\ F_{xy} \mid C{=}c &\sim \text{Categorical}(\alpha_c) \\ F_o \mid C{=}c &\sim \text{Normal}(\mu_c, \sigma_c). \end{aligned}$$

Here, $\text{Categorical}(\alpha_c)$ refers to a two-dimensional distribution with bin probabilities α_c. We have chosen to model the conditional probability $\mathrm{p}(F_o \mid C)$ using a normal distribution for practical reasons. Strictly speaking, using a normal distribution is an improper assumption because its support is $(-\infty, \infty)$ and not $[0, 1]$. However, probabilities beyond $\pm 3\sigma$ quickly drop to zero and therefore do not pose a problem for our use case.

The parameters of our model are estimated in a semi-supervised fashion. That is, we consider $\mathcal{F}$ to be observed for all training samples. In addition, we observe C for a fraction of training samples. The parameters of each distribution are estimated by maximizing the joint likelihood of fully observed and partially observed samples using a variant of expectation maximization (see Appendix A for details).

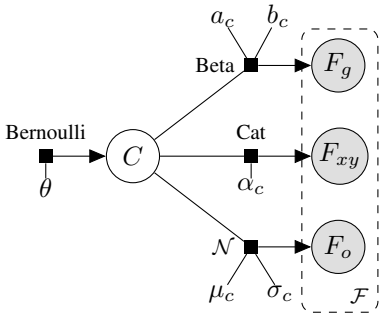


Fig. 5. Probabilistic data fusion. The presence of a command intended for robot r is modelled as a binary latent random variable C. Our model jointly considers a) gesture device confidence F_g, b) position of human F_{xy} and c) orientation of operator F_o to predict the posterior probability of C.

III. EVALUATION

A. Experimental Setup

We perform all experiments in an environment that covers the volume of $10 \times 8 \times 3.5\,\mathrm{m}$ using a single color camera with resolution $2464 \times 2056\,\mathrm{px}$, mounted $3.5\,\mathrm{m}$ above ground. We estimate the ground plane homography using a chessboard object. Our setup consists of two robots: an UR10 and a KUKA iiwa, both of which are placed in close proximity to each other (see Figure 7a). We use a standard computing unit equipped with a single NVIDIA GeForce GTX 1080 Ti for pose estimation at $15\,\mathrm{Hz}$. The gesture sensing device is connected to a portable Laptop. ZeroMQ[3] is used to exchange messages between all computing entities.

B. Pose Estimation Accuracy

In this work we consider metric accuracies, as image accuracies have been reported previously [6], [8]. Assuming an error-free intrinsic camera matrix, the uncertainty in detecting point correspondences for homography estimation is measured to be $\sigma_C = 3.5\,\mathrm{px}$. The parameter uncertainties of $\mathbf{H}_P^G$ are estimated using the method of Criminsi et al. [10]. The uncertainty of 2D keypoint detection is $\sigma_P = 15\,\mathrm{px}$. We transform input uncertainties to measurement uncertainties by propagating errors through Equation 1 to the first order. Table I lists measurement uncertainties as a function of object-to-camera distance.

Distance	Uncertainty
$3\,\mathrm{m}$	$0.05\,\mathrm{m}$
$10\,\mathrm{m}$	$0.30\,\mathrm{m}$

TABLE I

MEASUREMENT UNCERTAINTIES AS A FUNCTION OF OBJECT DISTANCE.

C. Classification Accuracy

To evaluate the data fusion approach, we conduct several experiments using the robot orchestration use case (see Section IV). For training we record 4500 observations of $\mathcal{F}$. This

[3] https://zeromq.org/

corresponds to 5 min of possible human-robot interaction. In a downstream step, we manually label C based on visual inspection. We then train several types of classifiers assuming a fully observed (FO), i.e. $\mathcal{F} \cup \{C\}$, to partially observed (PO), i.e. just $\mathcal{F}$, ratio of 2 %. Figure 6 shows the advantages of our proposed naïve Bayes method with semi-supervised training compared to a) classical naïve Bayes, b) SVM and c) a neural network. All classifiers are evaluated on a separate fully annotated test set of 5000 samples.

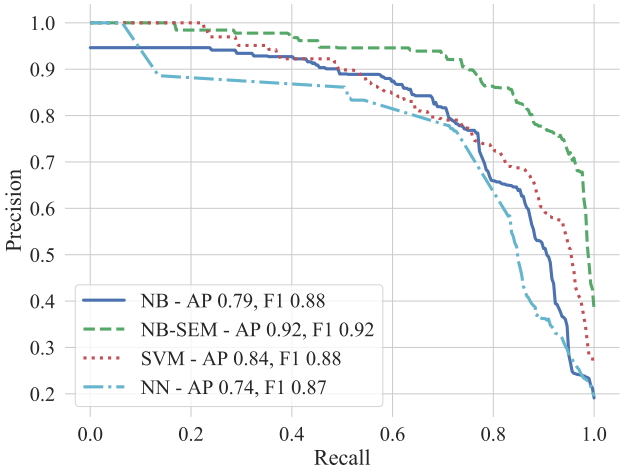


Fig. 6. Gesture classification performance. Each curve represents the precision/recall of one of the following classifiers: naïve Bayes (NB), naïve Bayes trained with unlabelled data (NB-SEM), support vector machine (SVM), neural network (NN). All classifiers are trained on 2 % of annotated data. Only NB-SEM additionally makes use of unlabeled data. Also shown: Average Precision (AP) and macro F1 score.

IV. DEMONSTRATIONS

We demonstrate the interaction potential of our system in two use cases (UCs)[4]:

- UC1 We orchestrate multiple robots by placing a near-field gesture-sensitive device in the hands of an operator. The operator's position and orientation with respect to the robots is used to predict the intended receiver of gesture commands (see Figure 7 a,b).
- UC2 We show an adaptive robot speed control to simplify human-machine cooperation. As humans approach, we automatically decelerate the robot. Likewise, we accelerate the robot to full operating speed as humans leave (see Figure 7 c,d).

V. CONCLUSION AND DISCUSSION

We present a single monocular camera-based system to provide real-time pose detection for human-machine interaction at large scales. Albeit our system does not sense depth directly, we develop a homography based solution to lift pose predictions to a metric 3D space. Furthermore, we show that merging global vision context and proximity data in a probabilistic framework helps to maintain robust detection over long distances. Finally, we present the results of a semi-monitored learning approach to data fusion that increases classification accuracy when only a few marked training samples are available.

[4]Video link is provided in footnote 1.

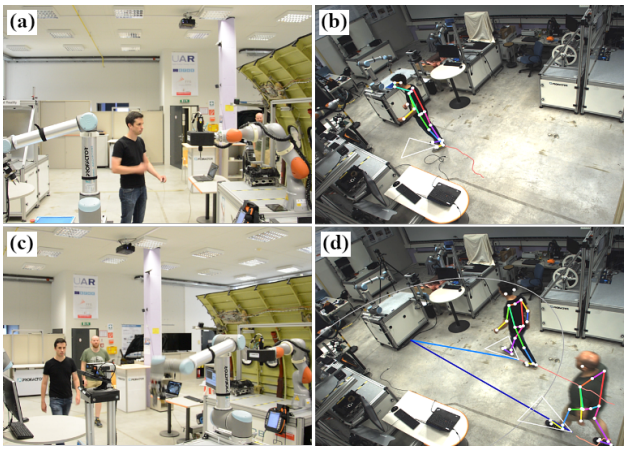


Fig. 7. Use Cases (UCs); (a,b) UC1 Robot orchestration - operator's position and orientation selects the intended robot to receive near-field gesture commands; (c,d) UC2 Robot Interaction - adjusting robot speed in human presence for near-field interaction.

REFERENCES

[1] Dana Hughes, John Lammie, and Nikolaus Correll. A robotic skin for collision avoidance and affective touch recognition. *IEEE Robotics and Automation Letters*, 3(3):1386–1393, 2018.

[2] Guanglong Du, Mingxuan Chen, Caibing Liu, Bo Zhang, and Ping Zhang. Online robot teaching with natural human–robot interaction. *IEEE Transactions on Industrial Electronics*, 65(12):9571–9581, 2018.

[3] Christian Zimmermann, Tim Welschehold, Christian Dornhege, Wolfram Burgard, and Thomas Brox. 3D human pose estimation in RGB-D images for robotic task learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1986–1992. IEEE, 2018.

[4] Flacco Fabrizio and Alessandro De Luca. Real-time computation of distance to dynamic obstacles with multiple depth sensors. *IEEE Robotics and Automation Letters*, 2(1):56–63, 2016.

[5] Petr Švarný, Zdenek Straka, and Matej Hoffmann. Versatile distance measurement between robot and human key points using RGB-D sensors for safe HRI. In *Proceedings of the 1st Workshop on Proximity Perception in Robotics at IROS*, 2018.

[6] Christoph Heindl, Thomas Pönitz, Andreas Pichler, and Josef Scharinger. Large area 3D human pose detection via stereo reconstruction in panoramic cameras. In *Proceedings of the OAGM Workshop*, 2018.

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[8] Christoph Heindl, Sebastian Zambal, Thomas Pönitz, Andreas Pichler, and Josef Scharinger. 3D robot pose estimation from 2D images. In *International Conference on Digital Image & Signal Processing*, 2019.

[9] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[10] Antonio Criminisi, Ian Reid, and Andrew Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.

[11] Christoph Heindl and Josef Scharinger. Notes on semi-supervised expectation maximization. *10.5281/zenodo.3484301*, October 2019.

[12] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

APPENDIX

A. Semi-Supervised Expectation Maximization

In semi-supervised training we assume that for a fraction of training samples we know the value of the latent variable C, while for a usually much larger fraction of data we don't. Following the notation introduced in Section II-D, we assume to be given N complete observations corresponding to the set $\{(C^i, \mathcal{F}^i)\}_{i \leq N}$ of random variables. In addition M partial observations of $\{\tilde{\mathcal{F}}^j\}_{j \leq M}$ are provided. All observations are considered to be independent and identically distributed. The model's global parameters are collected in $\Omega = \{\theta, a_c, b_c, \alpha_c, \mu_c, \sigma_c\}$. Figure 8 shows a graphical representation illustrating the situation.

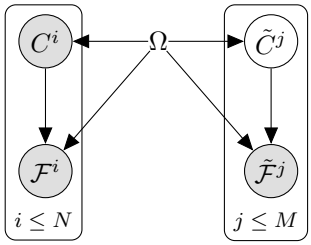


Fig. 8. Generative latent variable model with random variables grouped into fully and partially observed samples. Global model parameters are collected in Ω.

The joint probability given Ω factors as follows

$$p(\boldsymbol{C},\boldsymbol{\mathcal{F}},\tilde{\boldsymbol{C}},\tilde{\boldsymbol{\mathcal{F}}} \mid \Omega) = \prod_{i=1}^{N} p(C^i, \mathcal{F}^i \mid \Omega) \prod_{j=1}^{M} p(\tilde{C}^j, \tilde{\mathcal{F}}^j \mid \Omega). \tag{2}$$

where we denote $\{C^i\}_{i \leq N}$ by $\boldsymbol{C}$ and do similar for the other variables. We seek to optimize Ω by maximizing the log-likelihood of the observables

$$\Omega^* = \arg\max_{\Omega} \log p(\boldsymbol{C}, \boldsymbol{\mathcal{F}}, \tilde{\boldsymbol{\mathcal{F}}} \mid \Omega),$$

which requires integrating over latent $\tilde{\boldsymbol{C}}$. Following the objective of semi-supervised expectation maximization (EM) [11] gives

$$\begin{aligned}(\hat{q}, \hat{\Omega}) &= \arg\max_{q,\Omega} \mathcal{L}(q, \Omega) \\ &= \arg\max_{q,\Omega} \Big[\log p(\boldsymbol{C}, \boldsymbol{\mathcal{F}} \mid \Omega) + \\ &\qquad \mathbb{E}_{\tilde{\boldsymbol{C}} \sim q(\tilde{\boldsymbol{C}})} \log \frac{p(\tilde{\boldsymbol{C}}, \tilde{\boldsymbol{\mathcal{F}}} \mid \Omega)}{q(\tilde{\boldsymbol{C}})}\Big],\end{aligned} \tag{3}$$

where q is a tractable distribution over $\tilde{\boldsymbol{C}}$. Expectation maximization then iteratively optimizes for q (E-step) and Ω (M-step) in an alternating scheme, until a (local) maximum is reached. At iteration $t+1$, the solution to the E-step for our model is given by

$$\begin{aligned}q^{t+1}(\tilde{\boldsymbol{C}}) &= p(\tilde{\boldsymbol{C}} \mid \boldsymbol{C}, \boldsymbol{\mathcal{F}}, \tilde{\boldsymbol{\mathcal{F}}}, \Omega^t) \\ &= \prod_{j=1}^{M} p(\tilde{C}^j \mid \tilde{\mathcal{F}}^j, \Omega^t) = \prod_{j=1}^{M} q_j^{t+1}(\tilde{C}^j)\end{aligned} \tag{4}$$

where we made use of the independence assumptions underlying our model. The E-Step is thus equal to the original EM algorithm [12]. The M-step updates the parameters Ω of our model as follows

$$\begin{aligned}\Omega^{t+1} &= \arg\max_{\Omega} \mathcal{L}(q^{t+1}, \Omega) \\ &= \arg\max_{\Omega} \Bigg[\left(\sum_{i=1}^{N} \log p(C^i|\Omega) + \log p(\mathcal{F}^i|C^i, \Omega)\right) + \\ &\sum_{j=1}^{M} \sum_{c \in \{0,1\}} q_j^{t+1}(\tilde{C}^j = c) \log p(\tilde{C}^j = c, \tilde{\mathcal{F}}^j \mid \Omega)\Bigg],\end{aligned} \tag{5}$$

which follows from Equation 3 by applying the models independence assumptions and considering $q^{t+1}(\tilde{\boldsymbol{C}})$ to be constant. In Equation 5 the fully observed samples serve as a guidance bias for the optimization procedure. Finally, we solve for Ω^{t+1} by setting the gradient to zero

$$\nabla_{\Omega} \mathcal{L}(q^{t+1}, \Omega) = \mathbf{0}.$$

At `https://github.com/cheind/proximity-fusion` we provide source code for reproducibility.

B. Additional Classification Experiments

Table II compares the macro F1 scores of different classification approaches at varying fractions of full observations. Especially when only few annotated data points are available, our method outperforms the other classifier variants.

Fraction FO	NB	NB-SEM	SVM	NN
0.02	0.79	**0.92**	0.84	0.74
0.20	0.91	**0.93**	0.91	**0.93**
0.80	0.92	**0.94**	0.93	**0.94**

TABLE II

F1 MACRO SCORES OF GESTURE CLASSIFIERS AT VARYING LEVELS OF FULLY OBSERVED DATA (FO). SEE FIGURE 6 FOR ABBREVIATIONS.

2nd Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China

Extended Delta Compression Algorithm for Scanning LiDAR Raw Data Handling

Ievgeniia Maksymova [1,2], Christian Steger [2] and Norbert Druml [1]

***Abstract*— LiDAR sensors are widely used in robotic applications for depth image acquisition, and generate tens of frames per second that are transformed into a 3D point cloud. Each of these frames contains a pixel matrix of an entire field of view. In scanning LiDAR a frame is acquired sequentially and requires big on-chip memory arrays or a high speed interface for data to be transferred to ECU. In this paper we compare efficiency of existing lossless algorithms when applied to raw LiDAR data, and propose a lossless compression algorithm that is intended to reduce a required on-chip memory and/or relax speed requirement on an interface. The simulation results showed that achievable compression rate is $> 38\%$ independently from distance to target or receiving circuit resolution.**

I. Introduction

Due to advances and minituarization of the LiDAR technology, the sensor's application field has been expanded to robotics and unmanned aerial vehicles (UAV) [1]. LiDARs are widely used in mobile robots for a corridor mapping in a railway track inspections [2], pedestrian detection in urban scenarios [3], collision avoidance in hazardous and difficult accessible environments [4]. In recent years, also then automotive industry turned its attention to this technology in attempt to reach higher autonomous driving levels [5].

Along with an application field expansion, technical requirements of LiDAR sensors have also changed by addressing longer range ($> 200m$), higher resolution and frame rate, and lower production costs ($<$ \$200). A scanning LiDAR that uses MEMS mirrors for beam steering could potentially fulfill all aforementioned requirements. However, scanning long-range LiDAR sensors acquire frame data sequentially and require an intermediate memory before an entire frame can be processed into a 3D point cloud. The size of this memory is driven by the sensor's parameters such as range, resolution, an oversampling factor etc. [6]

3D point cloud computing is typically done using a complex DSP within the LiDAR sensor. This approach allows to produce results faster, requires big memory arrays for an intermediate data storage, custom implemented special functions and a long development cycle. Nevertheless, there are scenarios (e.g. sensor fusion) when raw data from several sensors shall be fused with the help of an external electronic computing unit (ECU). In this case the frame data shall be transferred from the sensor to ECU with a very high speed interface (Gb/s) and the internal memory array requirements could be relaxed. Both approaches have the same issue - high power consumption caused by either a big memory, or a very high speed interface.

In this paper we compare various low-level lossless compression algorithms that could be used in LiDAR sensors for a memory size reduction or improved bandwidth utilization. The algorithm comparison performed using several factors such as an implementation complexity, compression speed and effectiveness. Finally, a simple, yet effective compression algorithm is proposed that could be beneficial for battery-powered robots, e.g. UAV, and systems with a high frame rate requirements such as autonomous vehicles.

II. State-of-the-art

There are two main components in raw LiDAR data. First, an echo signal with a certain amplitude that depends on the target reflectance and the distance to a target. Second, a noise signal that can have several sources, e.g. an ambient noise or a shot noise of a receiving circuitry. Figure 1 shows two sets of measured LiDAR raw data that are done with different objects in the field of view with a distance up to 150 meters.

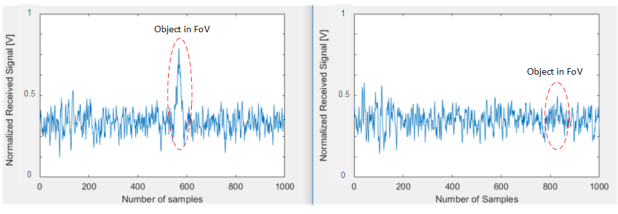


Fig. 1. Measured raw LiDAR data of targets with a high reflectance (left) and a low reflectance (right)

The dictionary-based compression algorithms require multiple iterations for building up a dictionary that later has to be saved or sent along with the compressed data for a successful data decompression. Thus, these algorithms are not effective for the LiDAR raw data compression.

[1]Infineon Technologies Austria AG {ievgeniia.maksymova; norbert.druml}@infineon.com

[2]Institute of Technical Informatics, TU Graz steger@tugraz.at

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation and Technology which funded the ODeLiA project under the grant agreement no 860032.

2nd Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China

Golomb-Rice Coding (GRC) compresses a data stream X of a length N using a tunable parameter M that is chosen to be a power of 2. The coding is performed by representing every value X_i as a sequence of 0's followed by 1 [7]. The number of 0s is defined by an integer result value of X_i divided by M. The reminder of this division is converted to $log_2(M)$ bits and adjusted to the 0's sequence.

In Delta Encoding (DE), the first value in a data stream is always written out the same as the original value. Following values are representing the difference between the current and the previous value in the data stream (delta). The codeword is S bits long to allow a delta ranging from -2^{S-1} to $2^{S-1}-1$. When a delta is too large to be represented by S bits, the original current value is stored as a codeword.

Symmetric Segmented Delta encoding algorithm (SSD) is a modified version of the delta encoding algorithm that computes deltas as the absolute value of the difference, which is then segmented in one of four segments [8]. Each of four segments has a defined $base$ that is used as a threshold for a proper segment placement, and an $offset$ that represents the reminder between the delta and the $base$. The output data is then stored as a delta sign bit, followed by a segment number and an offset.

As it can be observed in Figure 1, LiDAR data contains a DC component. Thus, removing this DC component through a careful selection of algorithm specific parameters (M, S, $base$ and $offset$) might improve resulting compression ratios, as less bits will be required to represent the same data.

III. Extended Delta Encoding Algorithm

The proposed Extended Delta Compression algorithm (EDC) is a derivative of the delta encoding algorithm that is used for data stream compression. The compression flowchart of EDC is shown in Figure 2. Every sample in the original data stream is a bw bit wide positive number. The maximum bit width reduction of a data stream sample bx is defined before the compression begins. The first value is always written out the same as the original value, while all computed delta values are compared with the 2^{bw-bx} value. In case a signed delta value is outside of the range, an overshoot is detected; the position of this overshoot is stored along with the full 2^{bw-1} bits wide delta value. Once all data has been encoded, the overshoot information is prefixed to the deltas as shown in Figure 3, where K is the number of occurred overshoots, $P-array$ are their positions, and $r-array$ are delta values.

The decompression flowchart is shown in Figure 4. First, the number of overshoots that occurred during compression (K) and their positions (P array) are extracted from the data stream. Second, the current sample's index is compared with values in P-array and the bit width of the current delta is defined. Then the difference between adjacent samples is calculated, resulting in lossless reconstruction of sampled data. As compressed data is a data stream, the pointer $plsb$ is used for tracking the position of the next sample and is incremented on a cycle base with the bitwidth of the current sample.

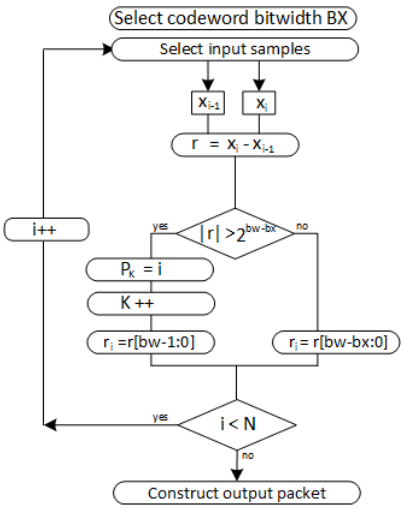


Fig. 2. Compression flowchart of the extended delta encoding

Fig. 3. Compressed data packet format.

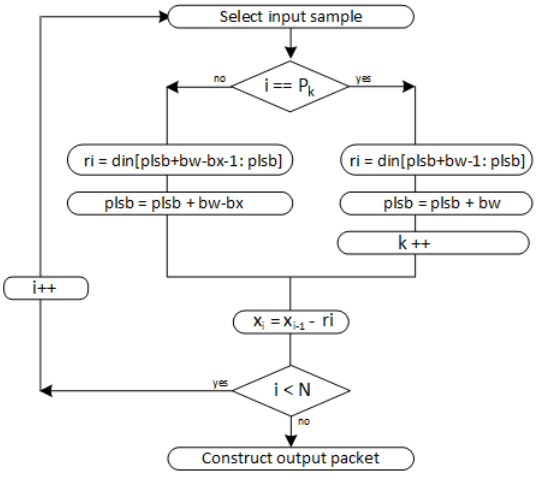


Fig. 4. Decompression flowchart of the extended delta encoding

IV. Simulation Results

Aforementioned algorithms are implemented in MATLAB and evaluated through a set of characteristics: compression ratio, compression speed, and algorithm complexity. The simulations were done using datasets acquired with a LiDAR demonstrator in a laboratory setup for collision avoidance. The following conditions were used: distance to a target ranging from 1m to 100m, weather factors set to no light,

bright light and rain, targets with reflectance levels ranging from 10% to 95%. The data has been digitized using ADC with two arbitrary ADC bit width values, 6 bits and 10 bits, to evaluate algorithms efficiency. Table I presents the evaluation results.

The compression ratio is used to evaluate algorithm performance and can be defined as

$$CompressionRatio = (1 - \frac{CompressedSize}{OriginalSize}) * 100\%$$

The speed of compression is evaluated in cycles required to complete the compression process of a data stream with N samples, assuming a cycle per operand. The complexity is defined by the amount of required hardware (e.g. buffers, adders, multipliers etc.) for an algorithm implementation.

TABLE I
COMPRESSION ALGORITHMS EVALUATION RESULTS

Algorithm	Speed in cycles	Complexity	C.Ratio ADC 6b	C.Ratio ADC 10b
DE	7*N	+++	19%	19%
GRC	7*N	++	-1.2%	1%
SSD	14*N	+	16%	28%
EDC	9*N	++	47%	38%

The Golomb-Rice Compression algorithm showed a very poor performance as the data DC value is far distanced from any of power of 2. The Delta Encoding showed a steady compression ratio independently from the data bit width. The Symmetric Segmented Delta encoding showed a tendency to have a better performance with high resolution data. The Extended Delta compression algorithm showed a good compression results ($> 38\%$) independently from the data bit width or the data set.

V. DISCUSSIONS AND CONCLUSIONS

In recent years, mobile robots, UAV and autonomous vehicles are more often using LiDAR sensors for depth imaging and range detection. LiDAR data acquisition, processing and transfer are resource demanding, which is a limiting factor for many compact and battery-powered applications. Through incorporating a raw data compression in a scanning LiDAR sensor, the sensor's memory demands could be optimized and complex high speed interfaces for data transfer between the sensor and the ECU could be eliminated. In this work the Extended Delta Compression algorithm was presented that is capable to compress LiDAR raw data by more than 38%, and the effectiveness of several low-level compression algorithms was compared.

As a continuation of this work, the proposed EDC algorithm will be used for a design optimizations of our LiDAR demonstrator, such as optimizing its interface bandwidth utilization. Moreover, as it has a good theoretical compression ratio and it is possible to calculate the maximum length of a compressed stream, it might be used to reduce on-chip memory needed to store a frame.

REFERENCES

[1] R. Moss, P. Yuan, X. Bai, E. Quesada, R. Sudharsanan, "Low-cost compact MEMS scanning LADAR system for robotic applications," Proc. of SPIE, vol.8379, 2012

[2] M.T. Andani, A. Mohammed, A. Jain, M. Ahmadian, "Application of LIDAR technology for rail surface monitoring and quality indexing," Journal of Rail and Rapid Transit, vol.232, 2018

[3] C. Premebida, O. Ludwig, U. Nunes, "LIDAR and vision-based pedestrian detection system," Journal of field of robotics, vol.26, 2009

[4] P.-L. Richard, N. Pouliot, S. Montambault , "Introduction of a LIDAR-based obstacle detection system on the LineScout power line robot," IEEE/ASME, ISBN: 978-1-4799-5736-1, 2014

[5] Corporate Partnership Board, "Automated and autonomous driving. Regulations under uncertainty," OECD/ITF 2015

[6] I. Maksymova, C. Steger, N. Druml, "Review of LiDAR Sensor Data Acquisition and Compression for Automotive Applications," Proceedings 2018, vol.2, pp.852-856, 2018

[7] P. Kalode and R. Khandelwal, "Test Data Compression based on Golomb Coding and Two-value Golomb Coding," SIPIJ, vol.3, no.2, pp.171-184, April 2012

[8] Shu-Ping Liang, Yi-Yu Liu, "Symmetric Segmented Delta Encoding for Wireless Sensor Data Compression," SASIMI Proceedings, 2016

3D Pose Estimation of Proximity Sensors with Self-Measurement for Calibration

Yitao Ding, Felix Wilhelm, and Ulrike Thomas

***Abstract*— The increasing number of sensing modules in a proximity servoing system for robotic applications requires new calibration methods. An exact pose calibration is essential for a correct obstacle detection. In this paper, we present a method for locating single proximity sensors on the surface of a robot based on the sensor's measurements of its environment, including self measurements of the robot. The algorithm relies on stochastic sampling methods to minimize the error between measured proximity data and simulated data by altering the poses of the simulated sensors. The simulation uses a virtual 3D reproduction of the robot and its environment.**

I. INTRODUCTION

This paper focuses on the localization of proximity sensors on the surface of the robot which can be used for the calibration of their poses. The calibration is gaining importance with increasing number of sensor modules, especially for flexible sensor skin designs, where an exact location after application is unknown. Accurate poses are in particular essential for a correct estimation of obstacles in 3D space. An error in sensor poses directly have negative impact on the sensor's forward kinematics and thus affects the calculated obstacle position in 3D space. The proposed method can be used for rough pose estimations and error detection of the system during operation. In combination with existing methods the accuracy and calculation time can be improved by either fine tuning the results of the existing methods or providing a starting point for optimization.

The localization of the sensor modules can be categorized in accurate positioning by design, manual calibration, and automatic calibration. Precisely manufactured mounting points on the sensors which fit to reference points on the robot have little deviation in their positioning. However, this method confines the flexibility of such systems and requires specially designed parts for different mounting locations on a robot. A precise pose calibration for randomly placed sensors on the robot can be performed manually by measuring the sensor position with a 3D measurement arm. This method delivers precise measurements but is time consuming, especially with higher numbers of sensing elements. Furthermore this task has to be repeated in full extend in case the sensor arrangement has been changed. Therefore, an automatic calibration method is needed to solve this problem. In the past, approaches have been shown based on external 2D/3D vision systems. In general, visual markers, such as AprilTags [1], provide pose information which can be used for localization when placed on the sensors. These markers require flat surfaces while the surface of the robot and sensors are often curved. Also, pose estimation algorithms for point cloud information [2] are a reasonable solution. In [3] an approach is presented which reconstructs the sensor poses from 2D images taken of the sensor's LEDs. However, the aforementioned methods rely on external sensing mechanisms which require calibration with respect to the robot frame. One solution to eliminate any further measurement equipment is to take advantage of the intrinsic information of the sensors and the robot system including the workcell. The authors in [4] exploit data from integrated acceleration sensors of the sensor modules and robot joint states to estimate the location with respect to their joint. In combination with the robot's forward kinematics the position on the link can be recovered.

Our similar approach only uses already available information. More precisely, the presented method works with time-of-flight information which are compared to data generated from a simulated environment. The environment is a virtual 3D reproduction of the robot and its surroundings where the robot's joints and sensor poses can be altered. Therefore, the estimation process uses the self-measurements of the robot and the workcell. In this process, a particle filter [5] generates different hypotheses of the virtual sensor locations, resulting in varying forward kinematics and thus creating different virtual proximity measurements in the virtual 3D space. The hypothesis with the smallest difference to the real measurement represent the most likely real world 3D location of the according sensor.

II. POSE ESTIMATION

A given proximity sensor with unknown location $\mathbf{p} \in \mathbb{R}^5$ (note: rotational invariance around the measurement direction of the sensors reduces one degree of freedom in orientation) on the robot's surface generates different measurement patterns $\mathbf{a} \in \mathbb{R}^k$ during k timesteps according to its location $\mathbf{p}$, the robot joint state $\mathbf{q}^n$ with n degree of freedoms (DOF), and the environment. Combined with the assumption of a static environment, these patterns are used for the estimation of the sensor location by comparison with patterns $\mathbf{a}^h$ of different location hypotheses $h \in [1, \ldots, m]$ generated in the virtual environment. This task can be formulated as an

All authors are with the Lab of Robotics and Human-Machine-Interaction at Chemnitz University of Technology, 09126 SN Chemnitz, Germany. Emails: {yitao.ding, felix.wilhelm, ulrike.thomas}@etit.tu-chemnitz.de

2nd Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China

optimization problem of:

$$\min\ Q(h) = \sum_{i=1}^{n} |a_i - a_i^h| \ . \tag{1}$$

The optimization structure (Fig. 1) includes a particle filter to avoid solutions in local minima and two stochastic optimization stages for finer local optimization. The series of k measurements reduces errors caused by sensor noise and inaccuracies in the 3D model. For example, objects in the real world unconsidered in the 3D model result in few outliers. Noisy inaccuracies of the models result in noisy readings which can be compensated with higher count of k. Of course our method fails when general offsets are present or when the percentage readings of inaccurate objects are large.

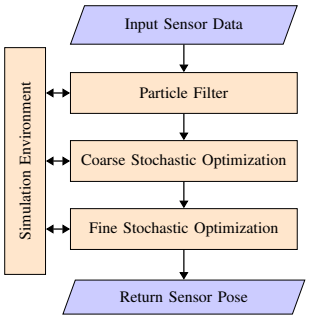


Fig. 1: Optimization Structure.

A. Particle Filter

Depending on the environment the sensor measurements $\mathbf{a}$ are not necessarily unique for all possible poses $\mathbf{p}$ and can be ambiguous. The particle filter for sensor pose estimation is a modification of a prior work [6]. Originally, it is used for pose estimation of grasped objects within a robotic gripper, in which ambiguous results (pose of grasped cylinder) can be described with the final distribution of the particles. While standard numerical optimization may converge into a local minimum, particle filters lower the likelihood due to their resampling process by randomly choosing different starting parameters.

Let $\mathcal{P}$ define a set of particles ρ of a particle filter with poses $\mathbf{p}(\mathbf{q})$ represented as transformation matrix controlled by $\mathbf{q}$ and their series of proximity measurements $\mathbf{a}$ performed at $\mathbf{p}$. First, we randomly seed w particles within the robot's surface hull approximated by a hollow cylinder for each link (Fig. 2). The orientation of these particles are most likely pointing in normal direction from the center of the cylinder. Thus, to obtain a normal distribution on a sphere the Bingham distribution [7] is used. With resampling, new particles ρ_{i+1} are picked randomly from $\mathcal{P}_i$ where lower cost Q particles have higher probability of being picked. Normal distributed noise is added afterwards. Particles ρ with Q above a certain threshold are uniformly distributed again. After the resampling process the particle distribution resembles the likelihood for a certain pose. Iterated resampling in combination of randomly reseeding particles refine the results and the particle with $\rho^{\min(Q(h))}$ approaches the global minimum.

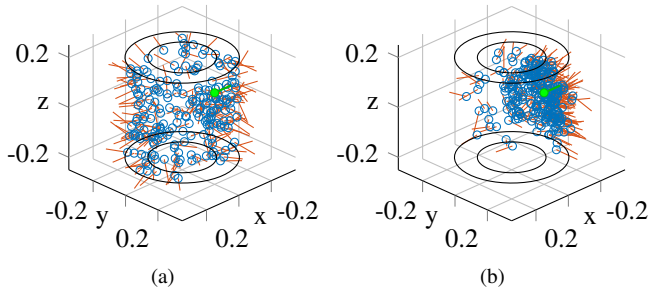


Fig. 2: Initial distribution of sensor poses within a link and distribution after first resampling. Green: real sensor pose.

B. Local Stochastic Optimization

The goal of the particle filter is to find a suitable starting point for local optimization. We maintain the particle nature of the particle filter in the local optimization step and iteratively reseed only $\rho^{\min(Q(h))}$ with w new particles with a Gaussian and Bingham probabilistic density function, which resembles a stochastic optimization method. The second stage has lower variance σ^2 for finer sampling and more accurate results.

C. Simulation Environment

We use V-REP [8] as simulation environment to generate virtual measurements which supports a variety of sensors including laser range sensors and capacitive proximity sensors. In the environment, robotic manipulators and 3D objects can be manipulated and combined to kinematic structures. The remote API for MATLAB (used in this work), Python and C provides easy integration into other systems. For this paper, the environment consist of a Kuka LBR iiwa 7 R800 and a ground plane to constrain measurements to a minimum of objects. Furthermore, in most cases the shape of the robot is known.

III. Results

To generate the reference signal a, we conducted real world measurements at 49 different joint configurations with a single sensor element placed on the end-effector. The estimations were performed with different amounts of particles $n = [100, 250, 500, 1000]$ with 40 estimations for each particle size. Each estimation consists of 5 iterations of the particle filter and 10 iterations of each local optimizer. Fig. 4 illustrates the progression of Q during the iterations. The search space is limited to a hollow cylinder with an inner radius of $0.15\,\mathrm{m}$, outer radius of $0.25\,\mathrm{m}$, and height of $0.4\,\mathrm{m}$.

2nd Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China

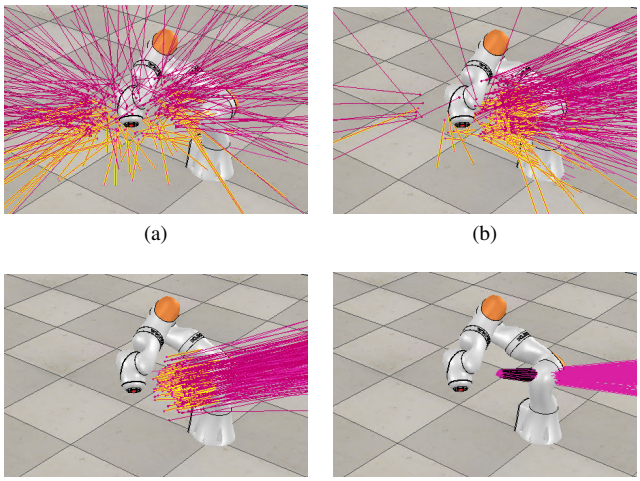


Fig. 3: Simulation environment in V-REP. a) Initial uniform distribution; b) after resampling; c) coarse sampling; d) fine sampling.

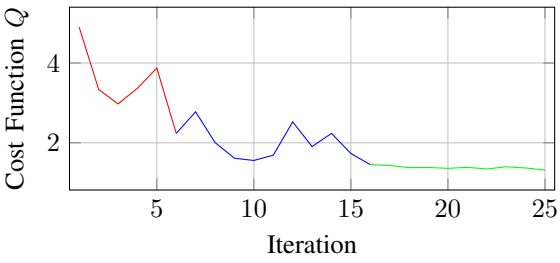


Fig. 4: Progression of Q with 1000 particles. Red: particle filter; blue: coarse sampling; green: fine sampling.

In Fig. 5 the mean absolute error (MAE) of the estimated position and orientation is shown with different amount of particles. The MAE is higher with 100 particles because the low number of particles result in local minima. With increasing particle numbers a reliable Euclidean MAE of below 20 mm is achievable. The lower MAE in x-direction along the measurement direction compared to y- and z-direction is caused by higher accuracy of the sensor readings, while poses in the other axis create similar readings. Furthermore the sensor position and the search space (Fig. 2) is limited by the thickness of the hollow cylinder and leads to a higher particle density in x-direction.

The orientation error is calculated with the reference direction vector $\mathbf{v}_\mathrm{r}$ and the estimated direction $\mathbf{v}_\mathrm{e}$. The error is then converted into degree representation.

$$e_\mathrm{Deg} = \arccos\left(\frac{\mathbf{v}_\mathrm{r}\mathbf{v}_\mathrm{e}}{|\mathbf{v}_\mathrm{r}|\,|\mathbf{v}_\mathrm{e}|}\right) \cdot \frac{180}{\pi} \qquad (2)$$

IV. CONCLUSIONS

The novel concept of using self-measurements for pose estimation is feasible. The advantage of independence means

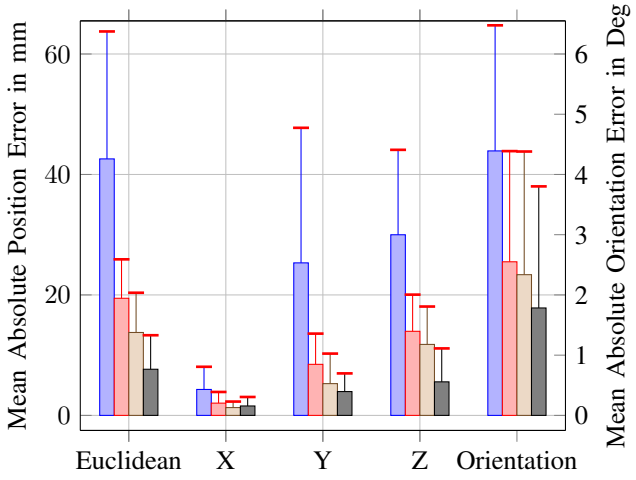


Fig. 5: Pose error after estimation with standard deviation. No. of particles: blue 100, red 250, brown 500, black 1000.

that estimation algorithm can be applied on any existing hardware without further external measurement equipment. The accuracy in position and orientation are not outstanding, but are in a reasonable range. The fact that we can achieve these results with simple random sampling based methods shows that there is potential for improvement which can lead to higher accuracy.

REFERENCES

[1] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[2] H. Kisner and U. Thomas, "Efficient object pose estimation in 3D point clouds using sparse Hash-Maps and Point-Pair features," in *50th International Symposium on Robotics (ISR 2018)*, Munich, Germany, Jun. 2018.

[3] P. Mittendorfer, E. Dean, and G. Cheng, "3d spatial self-organization of a modular artificial skin," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 3969–3974.

[4] E. Wieser, P. Mittendorfer, and G. Cheng, "Accelerometer based robotic joint orientation estimation," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, Oct 2011, pp. 67–74.

[5] S. Thrun, "Particle filters in robotics," in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 511–518.

[6] Y. Ding, J. Bonse, R. Andre, and U. Thomas, "In-hand grasping pose estimation using particle filters in combination with haptic rendering models," *International Journal of Humanoid Robotics*, vol. 15, no. 01, p. 1850002, 2018.

[7] C. Bingham, "An antipodally symmetric distribution on the sphere," *Ann. Statist.*, vol. 2, no. 6, pp. 1201–1225, 11 1974. [Online]. Available: http://dx.doi.org/10.1214/aos/1176342874

[8] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1321–1326.

[1] Karlsruhe Institute of Technology

[2] JOANNEUM RESEARCH ROBOTICS

[3] Inra Lille – Nord Europe

[4] Karlsruhe University of Applied Sciences

[5] Alpen-Adria-Universität Klagenfurt

[6] University of Tokyo Japan

Karlsruhe Institute of Technology
Intelligent Process Automation and Robotics Lab (IPR)
Engler-Bunte-Ring 8
76131 Karlsruhe, Germany
www.ipr.kit.edu

JOANNEUM RESEARCH ROBOTICS
Institute for Robotics and Mechatronics
Lakeside B13b
9020 Klagenfurt, Austria
www.joanneum.at

Inra Lille – Nord Europe
40 Avenue Halley
59650, Villeneuve d'Ascq, France
https://team.inria.fr/defrost/

Karlsruhe University of Applied Sciences
Moltkestr. 30
76133 Karlsruhe, Germany
www.hs-karlsruhe.de

Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt am Wörthersee, Austria
www.aau.at

The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo 113-8656, Japan
www.i.u-tokyo.ac.jp

Impressum

Karlsruher Institut für Technologie (KIT)
www.kit.edu