Hochschule Augsburg
University of Applied Sciences

Crazyflie quadcopter in decentralized Swarming

BAM days 2021

Klaus Kefferpütz, Thomas Izycki, Christos Zosimidis, Simon Zittenzieher

Hochschule Augsburg
University of Applied Sciences

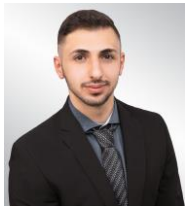
Cooperative Control Lab

- Faculty for Mechanical and Process Engineering, University of Applied Sciences Augsburg
- Objective: **Development and application of algorithms in the field of decentralized multi-agent systems**
- Crazyflie as primary hardware demonstrator
- Getting started in spring 2019
- We are happy to publish/share our results on github, website will be launched soon

Team members and projects

Klaus Kefferpütz

- Head of Cooperative Control Lab
- Professor for Measurement and Control
- Current project: **Crazyflie Navigation**



Christos Zosimidis

- B. Eng. in Engineering in Computer Sciences
- Embedded Software Developer at Engineering Office for Thermoacoustic (IfTA) GmbH in Puchheim, Munich
- Current project: **Decentralized communication**

Thomas Izycki

- B. Eng. in Engineering in Computer Sciences
- Master of Applied Research Study Program
- Current project: **Cooperative Path Planning**

Simon Zittenzieher

- B. Eng. in Mechanical Engineering
- Master of Mechanical Engineering Study Program
- Current project: **Exploration and Mapping**

Hochschule Augsburg
University of Applied Sciences

Outline

- Introduction – **Klaus**

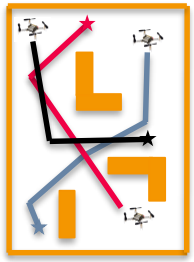
Part 1: Insights into current projects

- Cooperative Path Planning - **Thomas**
- Decentralized communication strategies for the Crazyflies – **Christos**
- First steps in (distributed) exploration/mapping – **Simon**

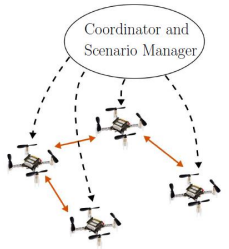
5 min break

Part 2: Navigation Enhancement for the Crazyflie

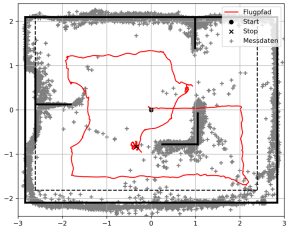
- Alternate Navigation Algorithm - **Klaus**



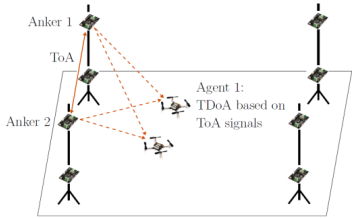
Cooperative Path Planning



Decentralized Navigation



Exploration and Mapping



Crazyflie Navigation/Localization

Hochschule Augsburg
University of Applied Sciences

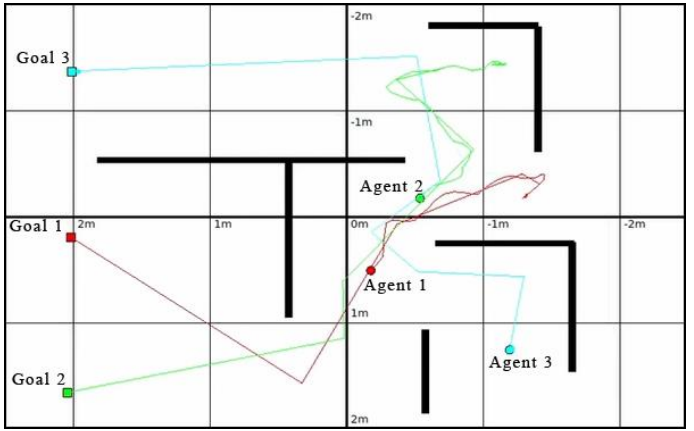
Cooperative Path Planning

Motivation

- Maneuvering through complex environments with a team consisting of multiple agents
- Individual execution of the path planning task to obtain a decentralized system

Cooperative Path Planning: Requirements/Challenges

- Implementation of a CL-RRT-based [1] onboard real-time path planning algorithm for the Crazyflie 2.0/2.1/Bolt
- Realization of the Merit-based Token Passing Coordination Strategy [2]
- Preference for agents with prioritized tasks

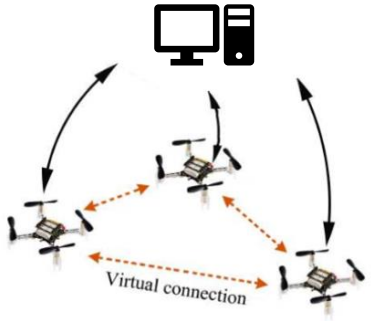


Cooperative path planning with three agents in an experimental setup

[1] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, “Motion planning in complex environments using closed-loop prediction,” AIAA Guidance, Navigation, and Control Conference (GNC), 08 2008.
[2] V. Desaraju and J. How, “Decentralized path planning for multi-agent teams with complex constraints,” Autonomous Robots, vol. 32, 05 2012.

Pseudo-Decentralized Communication

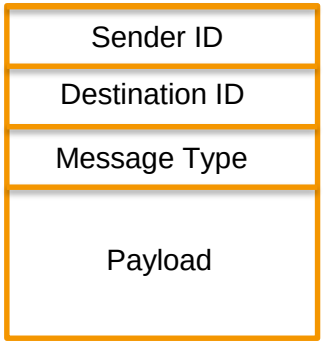
- Agents need to have the ability to broadcast messages and to exchange data directly between each other
- Messages get routed through the ground station



Pseudo-Decentralized Communication Framework

Interagent Communication for Path Planner Application

- Definition of a multi-agent packet
- Message types
  - Register/Remove team members
  - Add obstacles
  - Broadcast path
  - Broadcast bid
  - Pass on the token



Multi-Agent Packet

Coordination procedure

- First Step:
  - Agents plan their paths individually
  - The bid represents the improvement of the path
  - Agents with higher prioritized tasks can submit a higher bid
  - Collecting the bids is done by the current token holder

- Second Step:
  - The current token holder evaluates all submitted bids and passes the token to the winner
  - After receiving the token, the new token holder starts to follow the new calculated path and broadcasts the path to all team members
  - A new attempt to find a shorter path is started after the team constraints get updated

Rapidly-exploring Random Tree

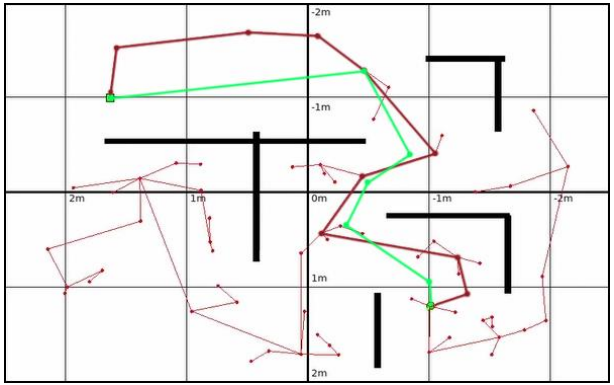
- Starting from the current position, randomly generated nodes are added to the tree based on obstacle feasibility
- Each new node is connected to the nearest existing node by a straight line
- The tree grows until a connection with the goal position could be established

Path Optimization

- Remove nodes if possible
- Move intermediate nodes closer

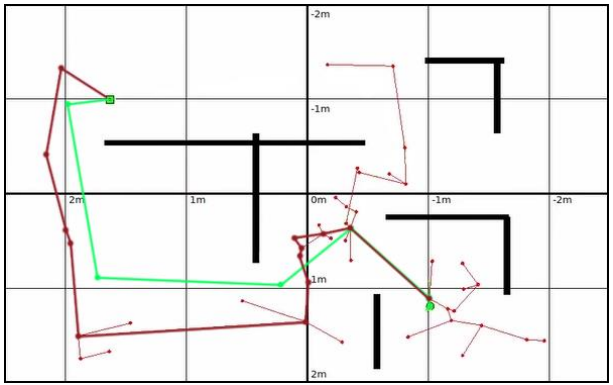
Continuous Planning

- The planning process is repeated
  - As long as no shorter path has been found
  - After every coordination iteration with the team while the goal has not been reached yet



Raw Path
Optimized Path

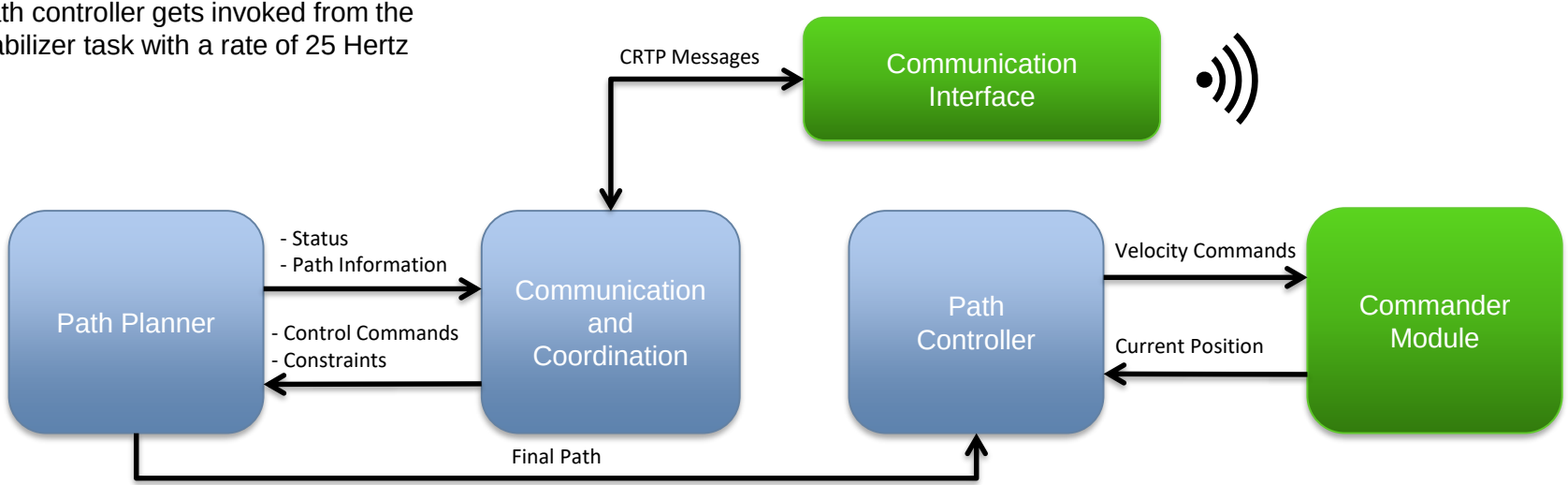
First feasible path from current position to the goal



Shorter path after further planning iterations

Integration into the Crazyflie Firmware:

- Path planner task runs with a low priority
- Communication and Coordination task runs with a medium/low priority
- Path controller gets invoked from the stabilizer task with a rate of 25 Hertz



Cooperative Path Planning in Action:
https://cloud.hs-augsburg.de/s/gkmyyCe4xJ2HyBK

Hochschule Augsburg
University of Applied Sciences

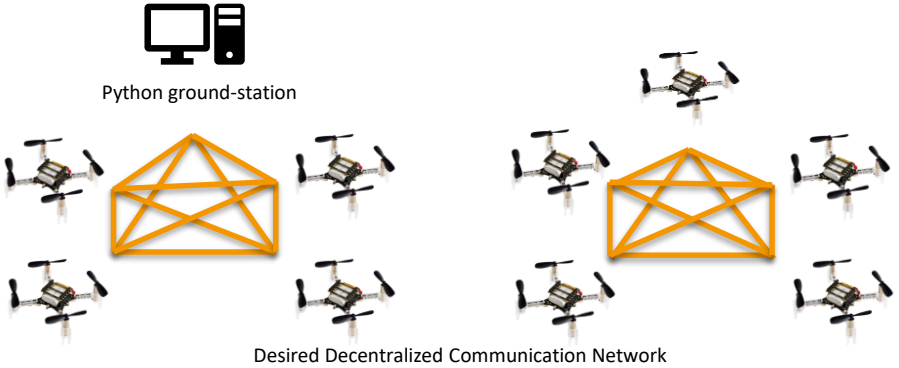
Decentralized Communication

Motivation

- Factory provided communication via Crazyradio PA is centralized, all communication is initiated and controlled by the Radio Dongle
- Decentralized methods for coordination and control require direct information exchange between the Crazyflie

Decentralized Communication: Requirements/Challenges

- Must allow for information exchange among the Crazyflies without involving the Crazyradio PA Dongle / Python ground station
- Communication with the Crazyradio PA Dongle / Python ground station should be possible but should **not be a "Must"**
    - Operator Control
    - Monitoring/Logging

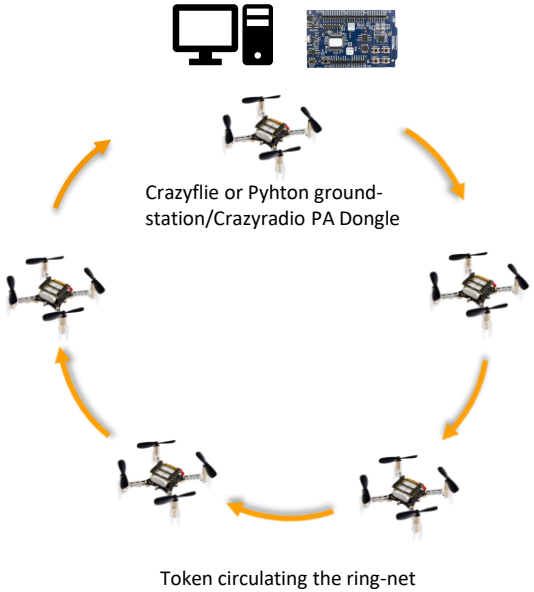


Evaluation of Communication Strategies

- Strategies relying on the detection of message collisions (e.g. CSMA) led to the Hidden-Node Problem [2] even for small numbers of Crazyflies
- Message collisions must be prevented

Token-Passing Strategy [1]

- All nodes are equal
- Ring network topology
- (Exaclty one) token circulating the network with high frequency
- Only nodes / Crazyflies in the possession of the token are allowed to transmit data
- Characteristics:
    - Decentralized, no need for central node to handle communication
    - Data collisions are impossible, only one node communicates at a time
    - No Hidden-Node Problem
    - Ideal for Swarm applications, as communication with each other is provided all the time
    - Fulfills real-time conditions
    - Network can be dynamically changed (e.g. auto-insert of nodes in the network / disconnect from the network)



Token circulating the ring-net

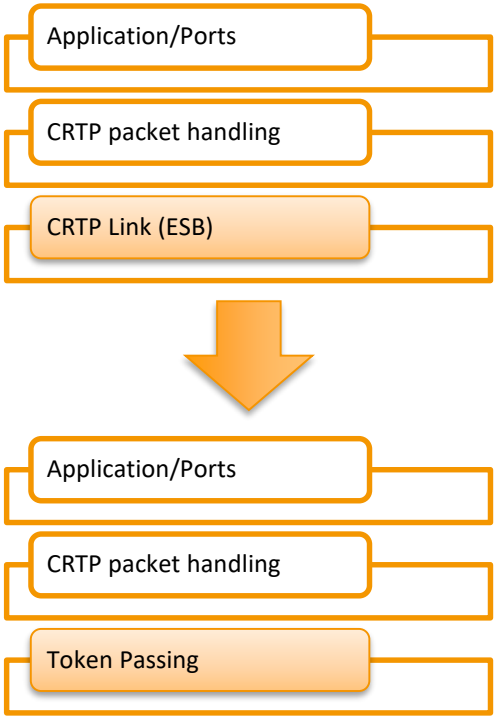
[1] M. Ergen, Duke Lee, Raja Sengupta and P. Varaiya, "WTRP - wireless token ring protocol," in IEEE Transactions on Vehicular Technology, vol. 53, no. 6, pp. 1863-1881, Nov. 2004, doi: 10.1109/TVT.2004.836928.
[2] Jangkeun Jeong ; Hyuntai Kim ; Sangtae Lee ; Jitae Shin: An analysis of hidden node problem in IEEE 802.11 multihop networks. (2010). – DOI https://ieeexplore.ieee.org/document/5573151

Implementation-Challenges

- Token-Handling: Token must never be lost or duplicated
  - Solved employing timers and additional messages
- Data-Handling: Ensure successful data transmission/prevent duplications
  - Solved employing acknowledgement frames/flags
- Auto-insertions in the network (work in progress)
  - periodically search for new nodes in the network
- Auto-disconnect (work in progress)
  - Signal other nodes the disconnect
  - Auto-throw from network if node fails to acknowledges the frames

Integration in Crazyflie Platform

- Use existent *Crazyflie Realtime Protocol (CRTP)* [2]
- Change link layer of CRTP from Enhanced ShockBurst (ESB) to Token Passing protocol [3]
- Even compatibility to Crazyflie-Client
- But now with decentralized communication

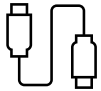


Original CRTP application port still available (logging, console etc.)

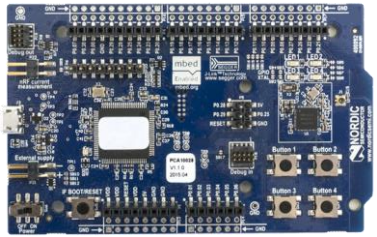
[2] Bitcraze AB: Crazy Real Time Protocol. 2021 https://wiki.bitcraze.io/projects:crazyflie:firmware:comm_protocol
[3] Nordic Semiconductor: nRF51 Series Reference Manual. 2016. – 85–88 S. https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v11.0.0%2Fesb_users_guide.html

Hochschule Augsburg
University of Applied Sciences

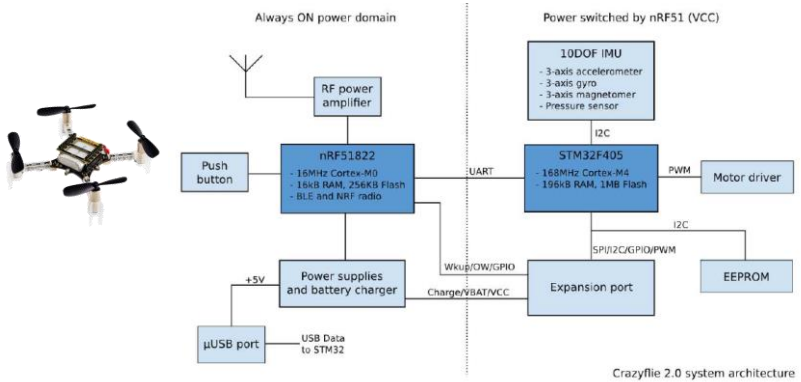
Implementation details



USB connection



nRF51 Developer kit replacing
Crazyradio PA Dongle



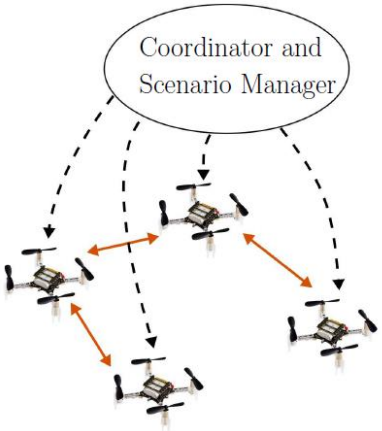
Crazyflie Architecture borrowed from www.bitcraze.io

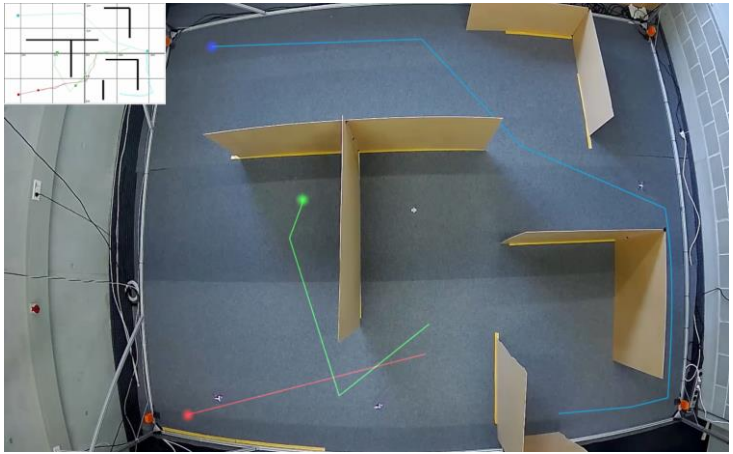
Python Ground-station/Crazyflie Python API

- cflib: CRTP-Driver for nRF51 Developer Kit
- COM-port: UART with syslink protocol as employed on Crazyflie
- nRF51-DK: Replacement for Crazyradio PA radio dongle
    - Communication with Crazyflie via radio 2.4 GHz
    - Employing Token-Passing Protocol

Crazyflie 2.0/2.1/Bolt

- Modified nRF51 firmware
- Use Token Passing protocol instead of ESB
- Modified radio link layer





Outlook

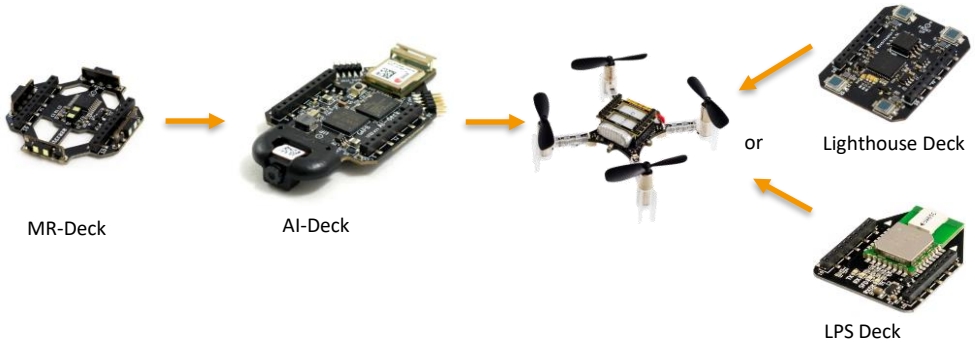
- Currently developing additional features and improving the implementation
- Performance evaluation / comparison with a centralized communication mimicking decentralized communication
- Integration in cooperative path planning application replacing pseudo-decentral communication
    - Remark: Already successfully demonstrated with pre-version of cooperative path planner
- We are happy to share the source code (Open Source with an MIT license)

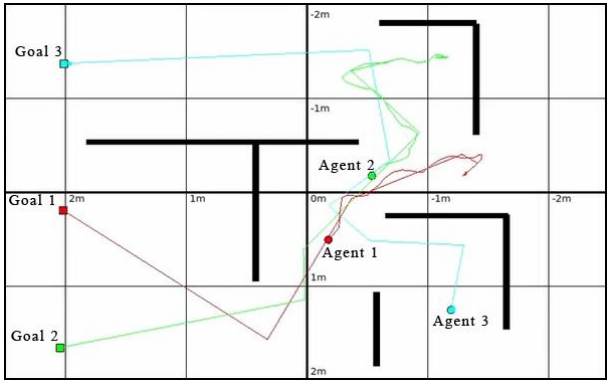
Hochschule Augsburg
University of Applied Sciences

(Decentralized) Exploration/Mapping

Autonomous Exploration and Mapping

- Cooperative path planning relies on a map of the environment which has to be provided manually
- Long-Term objectives:
  - Explore the environment autonomously with a Crazyflie team equipped with sensor decks (Multi-Ranger)
  - Form a common map based on the collected data which is shared through the network
  - Overcome restrictions due to resource constrained STM32F405 MCU making use of the AI-deck





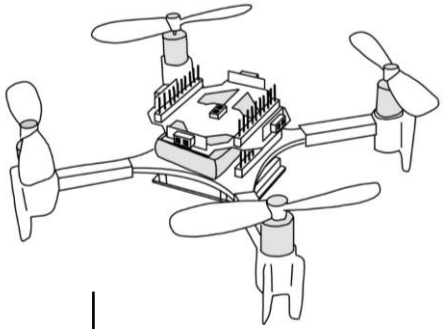
Cooperative path planning with three agents that know the position of the obstacles they are avoiding

First step:
Autonomous Exploration and Mapping with a single Crazyflie

Autonomous Exploration and Mapping with a single Crazyflie

- Goal: Enable drone to **autonomously explore** and gather information about an **unknown environment** and **extract obstacle coordinates**
- Exploration/Mapping algorithms
  - State-of-the-art SLAM methods not applicable due to low computational resources
  - Swarm Gradient Bug Algorithm (SGBA) already proved to be suited for exploration in [1], however, no map was created
  - To benefit from the AI-Deck and its toolchain, we decided to employ machine learning methods to extract obstacle data from MR point cloud measurements (map data compression)

- Approach:
  - Employ Swarm Gradient Bug Algorithm (SGBA) for exploration
  - Communicate MR-measurements and navigation data to a PC to register point-cloud data in a global map
  - Employ a machine learning algorithm to extract obstacle information
  - Reduction of complexity by assuming rectangular obstacles in rectangular shape



- Exploration algorithm
- Distance sensors
- Positioning system
- Communication (to external computation unit)

[1] McGuire, K. N. ; De Wagter, C. ; Tuyls, K. ; Kappen, H. J. ; Croon, G. C. H. E.: Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. In: Science Robotics 4 (2019), Nr. 35. https://robotics.sciencemag.org/content/4/35/eaaw9710.

Hardware and Architecture

Multi-Ranger Deck

Laser Distance Sensor

Lighthouse Positioning System

- Lighthouse Base-Station
- Lighthouse Deck

Loco Position System

- Used for comparison, judge influence of navigation quality on map

distances (front, back, left, right, down)

Lighthouse (Sweep Angle Measurements)

[Alternative: LPS & Flow Deck]

Crazyflie 2.1

- Bitcraze Firmware
- Modified wall-following algorithm [2] based on SGBA [1]

Modified wall following algorithm

Navigation Data-Fusion

position, attitudes

Crazyradio

Computer & Crazyradio Dongle

Motion commands

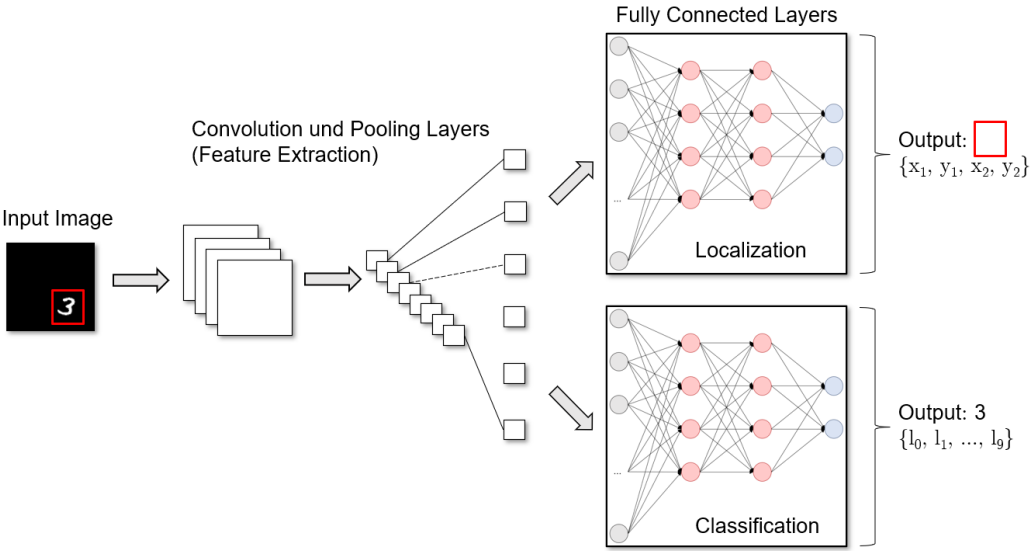
distances (front, back, left, right, down)

Registration of sensor data in global system
→ Machine learning application computes obstacle information

[1] Bitcraze crazyflie-firmware Github: https://github.com/bitcraze/crazyflie-firmware/tree/master/examples/demos/app_wall_following_demo/src
[2] McGuire, K. N. ; De Wagter, C. ; Tuyls, K. ; Kappen, H. J. ; Croon, G. C. H. E.: Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. In: Science Robotics 4 (2019), Nr. 35. https://robotics.sciencemag.org/content/4/35/eaaw9710.

Convolutional Neural Networks (CNN)

- Objective: Extract obstacle parameter (position, length, width) from point-cloud measurements
- Approach inspired by [3]
- Long term objective: Implement approach on AI-Deck for online processing



Network architecture

Python Implementation on ground-station

Long term objective

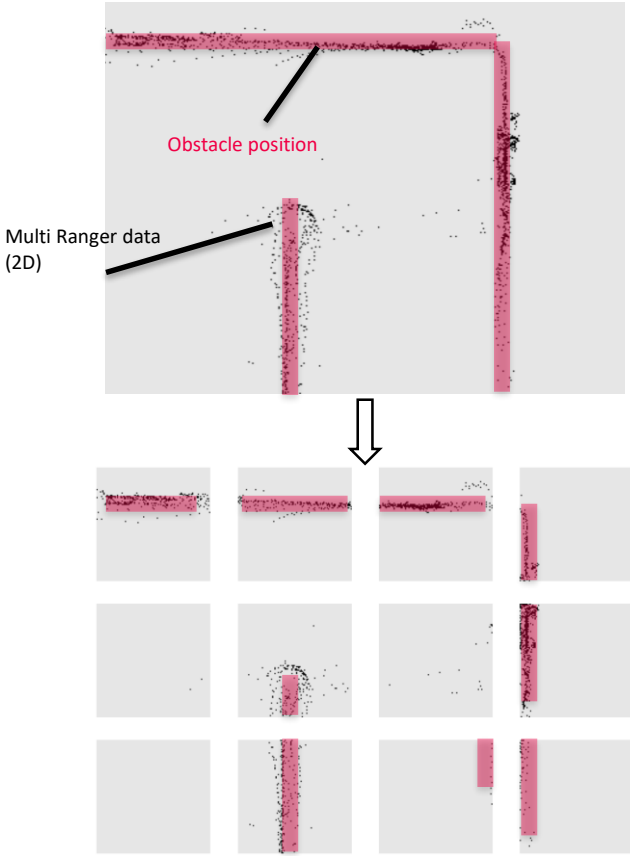
AI Deck

[3] Thakur, A.: Object Localization with Keras and WandB. https://medium.com/analytics-vidhya/object-localization-with-keras-2f272f79e03c/

Hochschule Augsburg
University of Applied Sciences

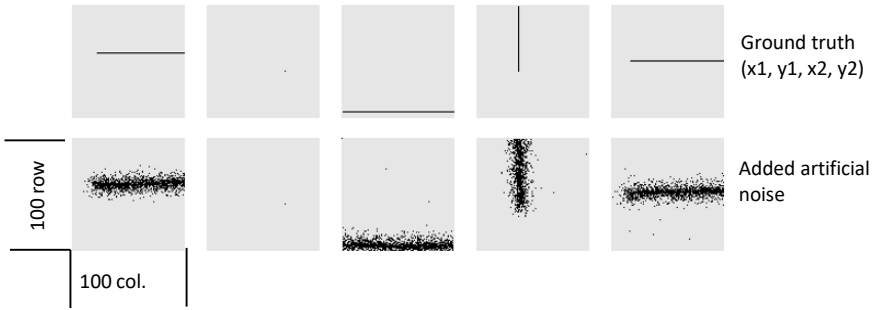
Convolutional Neural Networks – Application for measuring points?

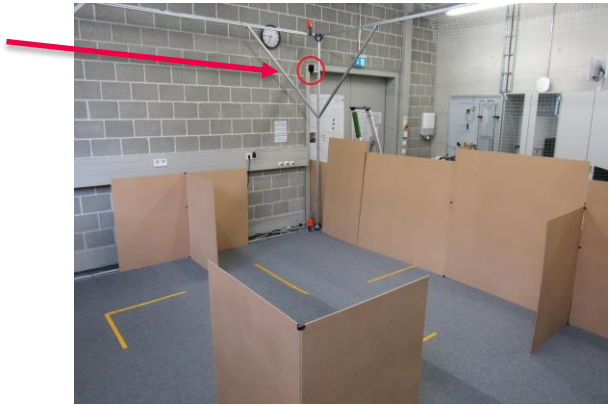
- CNN outputs only single classification / localization-output for every prediction of the artificial network
  - Separation of data in quadratic input-images
  - Step-by-step prediction and storage of classification and localization output
  - Combination of multiple obstacles

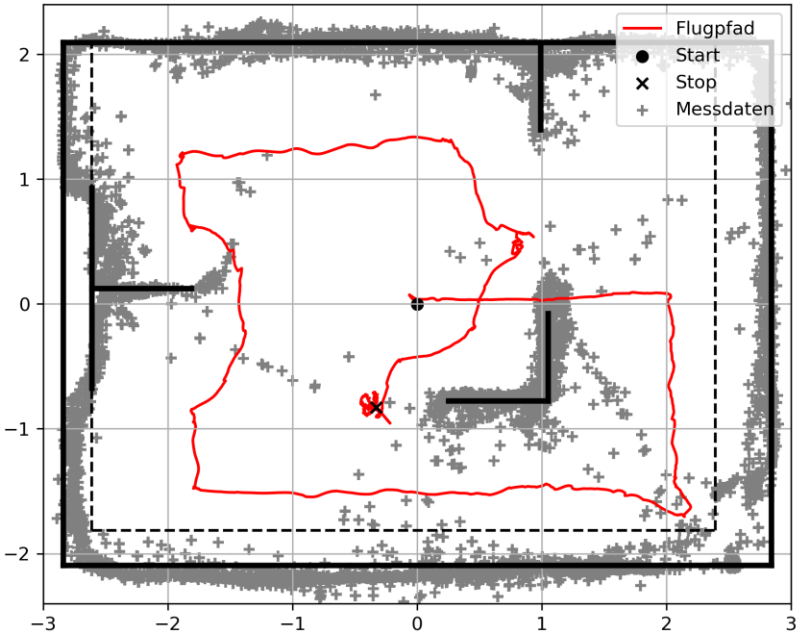


Training and Artificial training data

- CNN accepts only one predefined input-format (here: 100x100)
- Training with artificially generated measurement data due to lack of real data (50’000 images for training with ground truth data)







Laboratory set up

- 4 m x 5 m indoor set up with walls and additional obstacles
- Positioning with Lighthouse System
  (2x Lighthouse base-stations)

Results

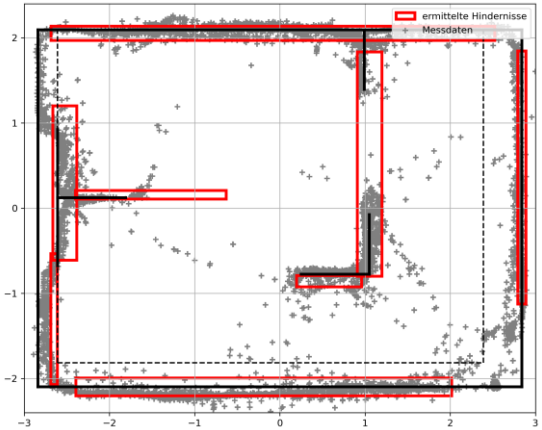
- Time of flight: ~ 118 seconds
- All obstacles covered
- Some outliers, mostly at sharp edges

Video Point Cloud
https://cloud.hs-augsburg.de/s/trfesxykBFNij5J

Hochschule Augsburg
University of Applied Sciences

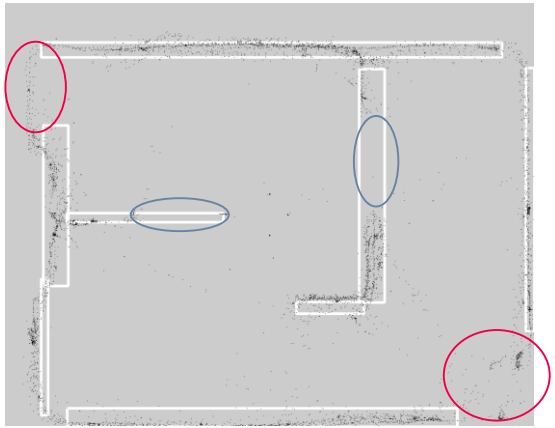
Interpretation of results

- Sufficient recording of environment
- But: weak coverage of corners (→ adjust / slow down yaw-motion)
- Outliers (on sharp edges) lead to enlarged output obstacles

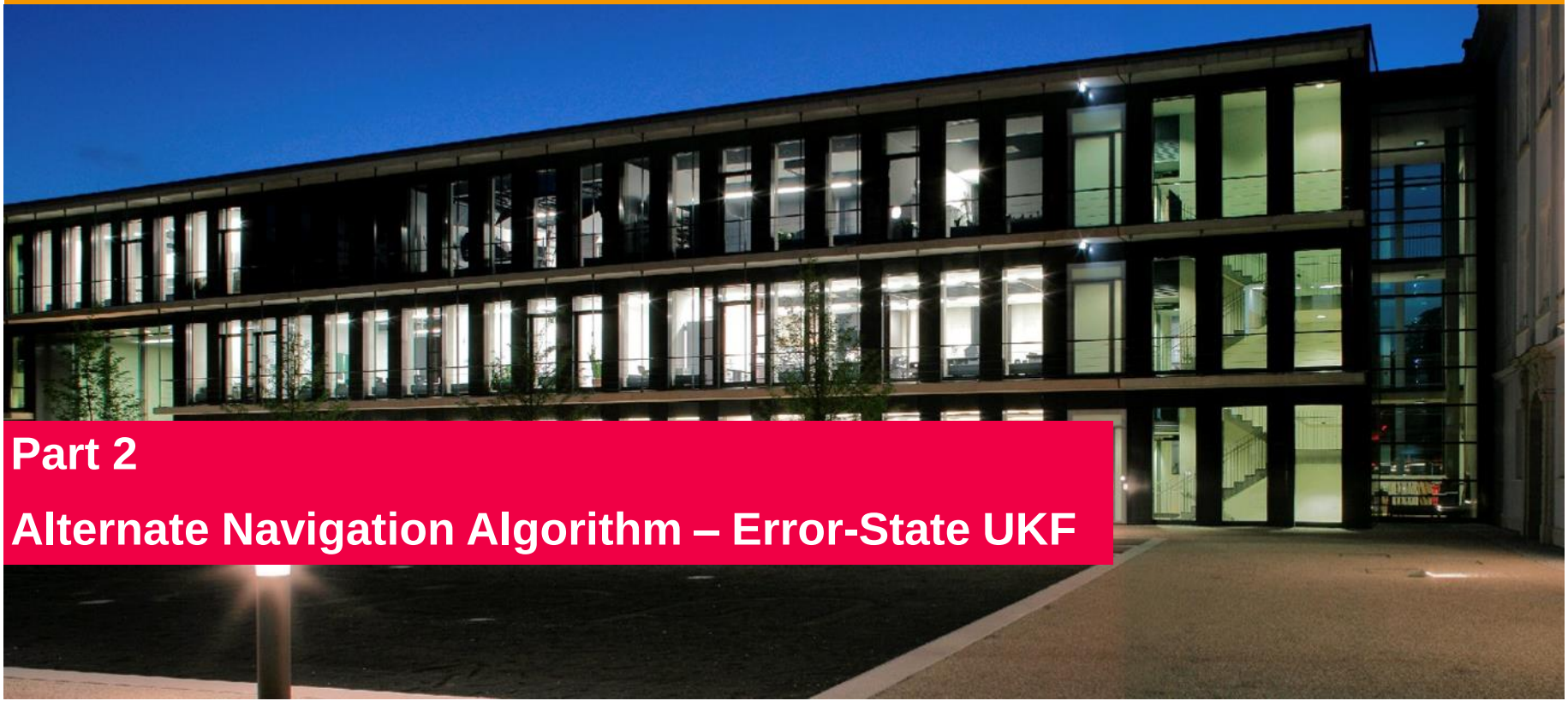


Outlook

- Optimization of exploration algorithm
  - Improve reliability / safety
  - Bias search direction towards unexplored areas and/or regions, where obstacle identification is uncertain
- Optimization of obstacle detection
- Extend approach to decentralized swarming
- Implement approach on the AI-Deck for onboard processing



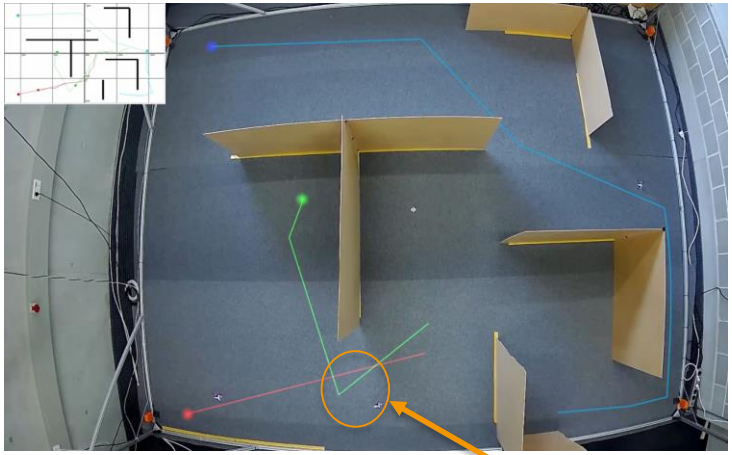
Hochschule Augsburg
University of Applied Sciences



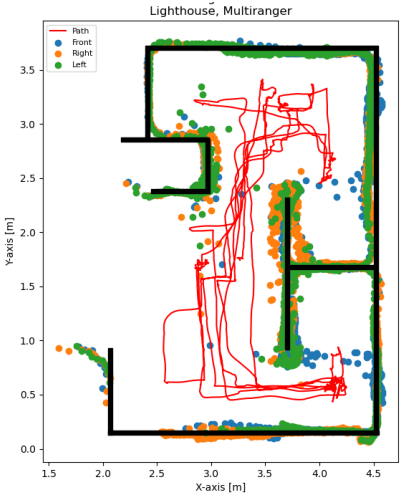
Part 2

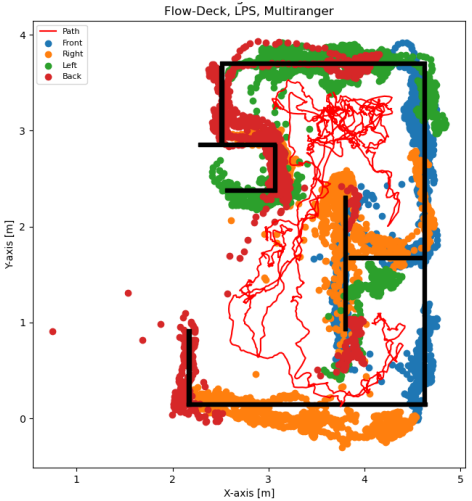
Alternate Navigation Algorithm – Error-State UKF

Alternate Navigation Algorithm



Position error when relying on LPS and Flow-deck v2





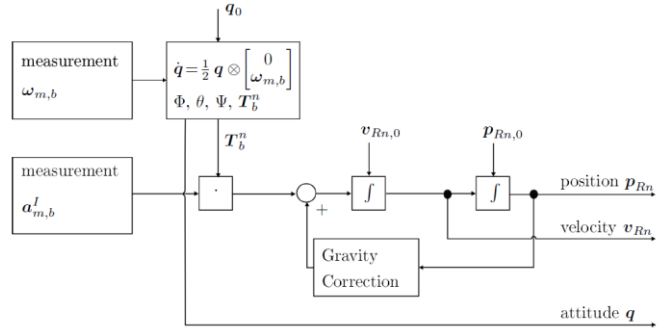
Crazyflie Positioning / Navigation

- Accurate positioning is an important requirement in most multi-agent scenarios
- Indoor positioning at the Cooperative Control Lab
  - Lighthouse System (LH): Sweep Angles of IR-lightplanes emitted of 2 spatially separated base-stations (accuracy 2-4 cm), requires direct line-of-sight contact to the base-stations
  - **Loco-Positioning System (LPS):** Time difference of arrival w.r.t. 8 spatially separated anchors (accuracy ~15 cm)
  - Flow-Deck: Relative positioning based on optical flow and time-of-flight measurements (primarily used as additional aiding sensor in LPS environments)

Our aim is improving navigation accuracy based on LPS and extending it towards cooperative navigation

Error-State Navigation Filter [3]

- Strapdown Algorithm [2]: Time integration of IMU measurements combined with
- Extended Kalman Filter (EKF) estimates the error-state based on available aiding sensors (LPS, Flow Deck v2,….)
  - Strapdown solution corrected with each measurement update
  - Error-state is set to zero after correction
- No wind-free assumptions, no drone specific parameters aside sensor covariances



Strapdown navigation, e.g. [2]

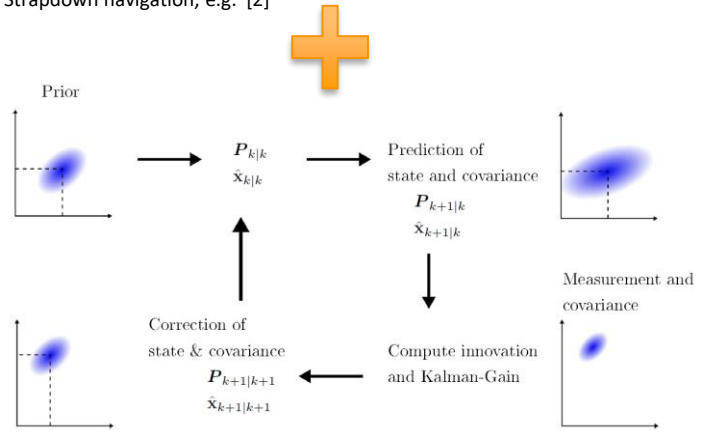
Extended Kalman-Filter (EKF)

- Aim is estimating the state and covariance of a nonlinear, stochastic system

$$\mathbf{x}_{k+1} = \boldsymbol{f}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\eta}_k$$
$$\mathbf{y}_k = \boldsymbol{h}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\zeta}_k$$

with $\boldsymbol{\eta}_k \sim N(\mathbf{0}, \boldsymbol{Q}_k)$, $\boldsymbol{\zeta}_k \sim N(\mathbf{0}, \boldsymbol{R}_k)$

- Nonlinearities are approximated, computing **Jacobians** in the prediction and measurement update steps



Kalman-Filter: Computational Steps

[1] M.W. Mueller, M. Hamer; R. D'Andrea: Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation, IEEE international Conference on Robotics and Automation, ICRA 2015
[2] D.H. Titterton, J.L. Weston, "Strapdown Inertial Navigation Technology - Second Edition", Institution of Electrical Engineers, 2004
[3] Sola - Quaternion kinematics for the error-state Kalman Filter, http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf, October 2017

Measurement Equations

- LPS / UWB (we solely consider TDoA mode 3)

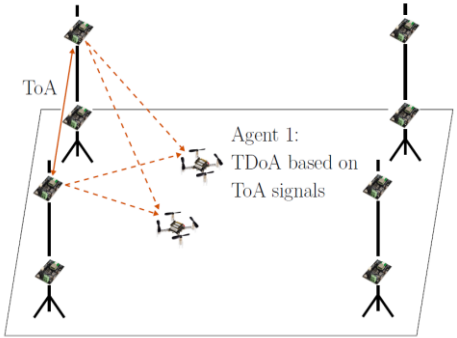
$$\hat{y}_{tdoa} = h_{tdoa}(\hat{\mathbf{x}}) = \sqrt{(\hat{x}_g - x_j)^2 + (\hat{y}_g - y_j)^2 + (\hat{z}_g - z_j)^2} - \sqrt{(\hat{x}_g - x_i)^2 + (\hat{y}_g - y_i)^2 + (\hat{z}_g - z_i)^2}$$

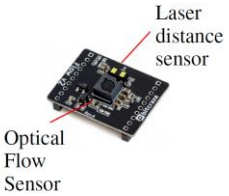
- Flow-Deck time-of-flight measurement

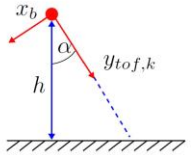
$$\hat{y}_{tof} = h_{tof}(\hat{\mathbf{x}}) = \frac{z_g + \Delta z_g}{\cos(\alpha)} = \frac{z_g + \Delta z_g}{r_{33}}$$

- Flow-Deck optical flow measurement

$$\hat{\boldsymbol{y}}_{flow} = \boldsymbol{h}_{flow}(\hat{\mathbf{x}}) = \frac{N\Delta t}{\theta_p} \cdot \begin{bmatrix} v_{b,x}\frac{r_{33}}{z_g+\Delta z_g} + \omega_{m,b,y} \\ v_{b,y}\frac{r_{33}}{z_g+\Delta z_g} - \omega_{m,b,x} \end{bmatrix}$$







Extended Kalman-Filter: Measurement Update Step

- Compute Jacobian of nonlinear measurement equation

$$\boldsymbol{H}_{k+1} = \left.\frac{\partial \boldsymbol{h}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}}_{k+1|k} \\ \mathbf{u}=\mathbf{u}_{k+1}}}$$

- Compute Kalman Gain

$$\boldsymbol{K}_{k+1|k} = \underbrace{\boldsymbol{P}_{k+1|k} \cdot \boldsymbol{H}_{k+1}^{\mathrm{T}}}_{P_{xy}} \cdot \underbrace{\left(\boldsymbol{R}_{k+1} + \boldsymbol{H}_{k+1} \cdot \boldsymbol{P}_{k+1|k} \cdot \boldsymbol{H}_{k+1}^{\mathrm{T}}\right)^{-1}}_{P_{yy}{}^{-1}}$$

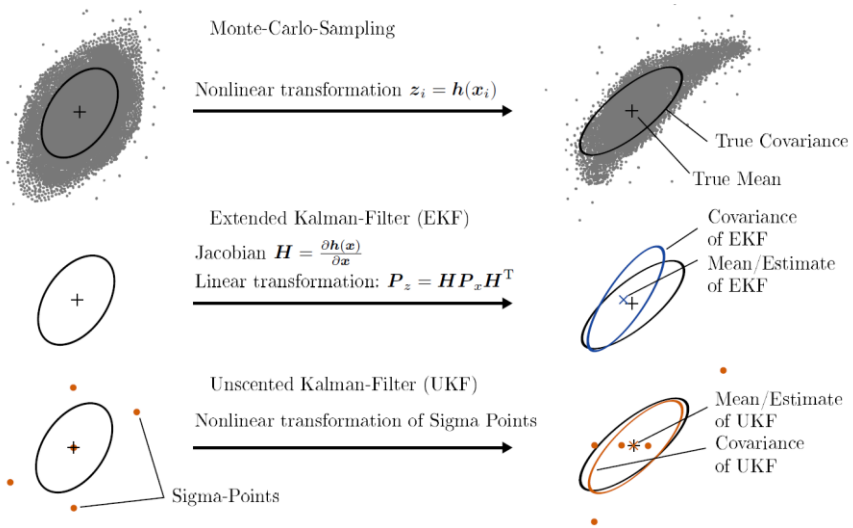
- Update state estimate:

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \boldsymbol{K}_{k+1} \cdot \left(\mathbf{y}_{k+1} - \hat{\mathbf{y}}_{k+1|k}\right)$$

Unscented Kalman-Filter (UKF):

- Aims on increasing the approximation quality of the covariances for state estimate and measurement
- Applies the unscented transformation based on so-called sigma-points to approximate the covariance instead of Jacobian computations
- The computational complexity is similar compared to the EKF

We combined the error-state navigation approach of [3] with the Unscented Kalman Filter [4]



Approximation abilities of the extended vs. the unscented Kalman-Filter

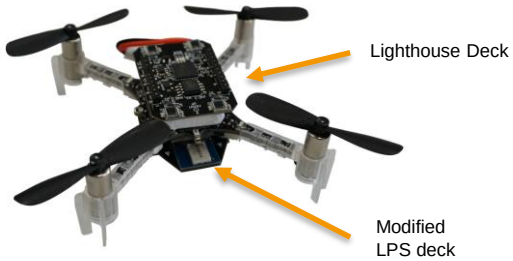
Fusing Flow-Deck measurements

- Fusing optical flow and height measurements improve the navigation accuracy
- However, flying over obstacles, other Crazyflies, etc. is causing undesired “jumps”
- We implemented a **straightforward outlier detection** based on the measurement covariance to prevent unlikely measurements from being processed by the UKF

[3] Sola - Quaternion kinematics for the error-state Kalman Filter, http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf, October 2017
[4] S.J. Julier; J.K. Uhlmann, “A New Extension of the Kalman Filter to Nonlinear Systems“, Signal Processing, Sensor Fusion, and Target Recognition VI, Aerosense 97, Orlando, USA, 1997

Accessing the LPS navigation quality

- Ground truth data is mandatory for parameter tuning and evaluation
- We employed the Lighthouse system with crossing beam method [5]
- Crazyflie 2.1 equipped with
  - Modified LPS deck (switched serial port)
  - Lighthouse Deck (not fused in the UKF)
- Lighthouse and LPS coordinate frames have to be aligned in a post-processing step before numerical evaluation (approach stated in [6])



Crazyflie 2.1: Configuration with Lighthouse Deck and (modified) LPS deck

Installation of LPS Positioning system in the lab

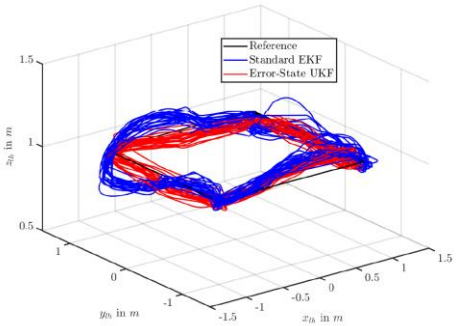
Improvised “Corona-Lab”

[5] A. Taffanel; B. Rousselot; J. Danielsson; K. McGuire; K. Richardsson; M. Eliasson; T. Antonsson; W. Hönig, “Lighthouse Positioning System: Dataset, Accuracy, and Precision for UAV Research“, ICRA Workshop on Robot Swarms in the Real World, Arxiv 2021
[6] K. S. Arun; T. S. Huang; S. D. Blostein, “Least-Squares Fitting of Two 3-D Point Sets“, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 9, No.5, 1987

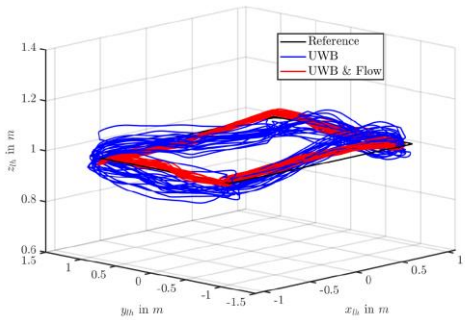
Hochschule Augsburg
University of Applied Sciences

Experimental Results

- Pure LPS based navigation
  - Navigation error reduced by ~ 20% w.r.t. standard EKF
  - Visually, also attitude stability is enhanced
  - A preflight height offset of ~30 cm is observed for standard EKF and error-state UKF
- Fusing LPS and Flow-Deck v2 measurements
  - Reduces the height estimation errors significantly
  - Outlier rejection successfully prevents height distortions when passing over obstacles



Trajectory plots for standard EKF and error-state UKF when relying on LPS



Trajectory plots for error-state UKF for pure LPS and LPS plus optical flow/time-of-flight measurements

Evaluate yourselves:

- Error-state UKF has been submitted to ICRA 2022
- Firmware has been published on github [7]
  - Please note, that default navigation is still standard EKF, you have to switch to error-state UKF prior to take-off
  - We also provided a measurement update function for Lighthouse sweep angle measurements, however
  - we do not have a Mocap system so there is no quantitative evaluation for Lighthouse sweep angle measurements

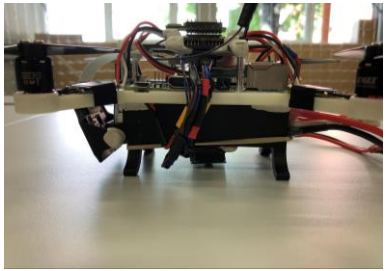
Video Error-state UKF vs. Standard EKF:
https://cloud.hs-augsburg.de/s/AFsQRFPP6faY35x

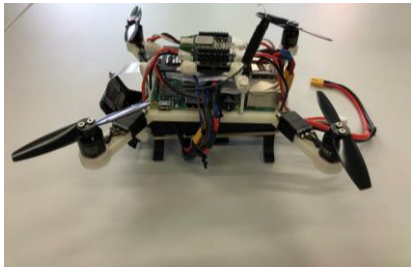
[7] Error-State UKF on Github (out-of-tree version):
https://github.com/HSaugsburgBitcraze/estimator_error_kalman

Hochschule Augsburg
University of Applied Sciences

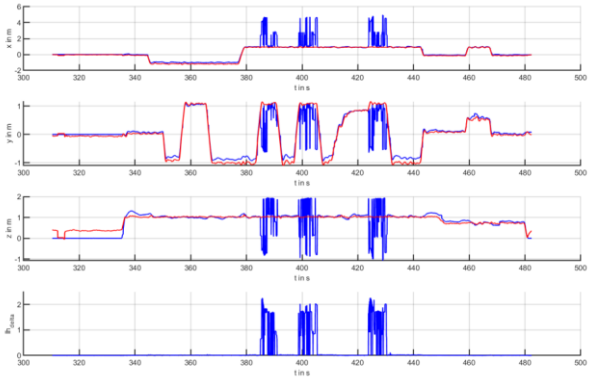
Potential Pitfalls

- Lighthouse System with crossing beam method
  - Prone to any reflective material, cover all windows
- Flow-deck v2 primarily when applied with standard EKF
  - Tends to diverge if clearance below the Flow deck is chosen too low,
  - Also happens with 2 decks below the Crazyflie (modified LPS and Flow Deck) or
  - when using long deck connectors



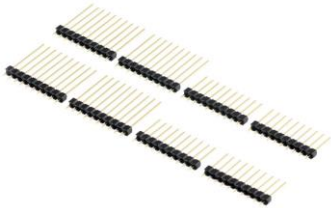


Prototype of Bolt based customized drone



Windows causing reflections distorting Lighthouse crossing beam method

Long deck connectors



Standard deck connectors

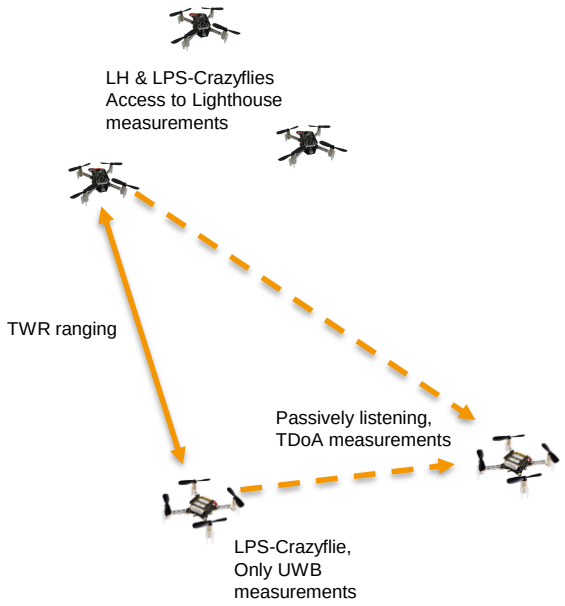
Cooperative Navigation

- Limited amount of slots for sensor decks on a Crazyflie, so data-fusion on each Crazyflie limits the abilities of the overall multi agent team
- LPS deck allows measurements of ranges in between Crazyflies, no additional sensor-deck is necessary, so
  - First steps towards “TDoA hybrid mode” already done by Bitcraze, so
  - Crazyflies with additional Lighthouse decks and/or
  - Crazyflies fusing LPS and Flow-Deck v2 measurements

  could serve as mobile anchors for pure LPS-Crazyflies

Challenges:

- Methodological
  - Due to inter-Crazyflie range measurements, estimated navigational states of different Crazyflies become correlated
  - Cross-correlations must be accounted for to prevent consistency issues (i.e. applying covariance intersection [8], [9])
- Implementation
  - Extend LPS communications to distribute the position estimate and covariance information among Crazyflies
  - Limited computational resources of the STM32F405 MCU



Concept of Cooperative Navigation

[8] S. J. Julier; J. K. Uhlmann: A Non-divergent Estimation Algorithm in the Presence of Unknown Correlations, Proc. Of the American Control Conference, June 1997
[9] N. Noack; J. Sijs; U. D. Hanebeck: Inverse Covariance Intersection: New Insights and Properties, Proc. of the Conference on Information Fusion, 2017

Thank you and

Birthday to the whole Bitcraze Team!