

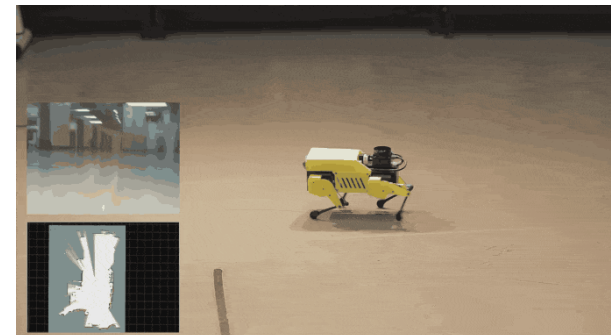
Multi-Sensor-Systeme

Robot Operating System (ROS)

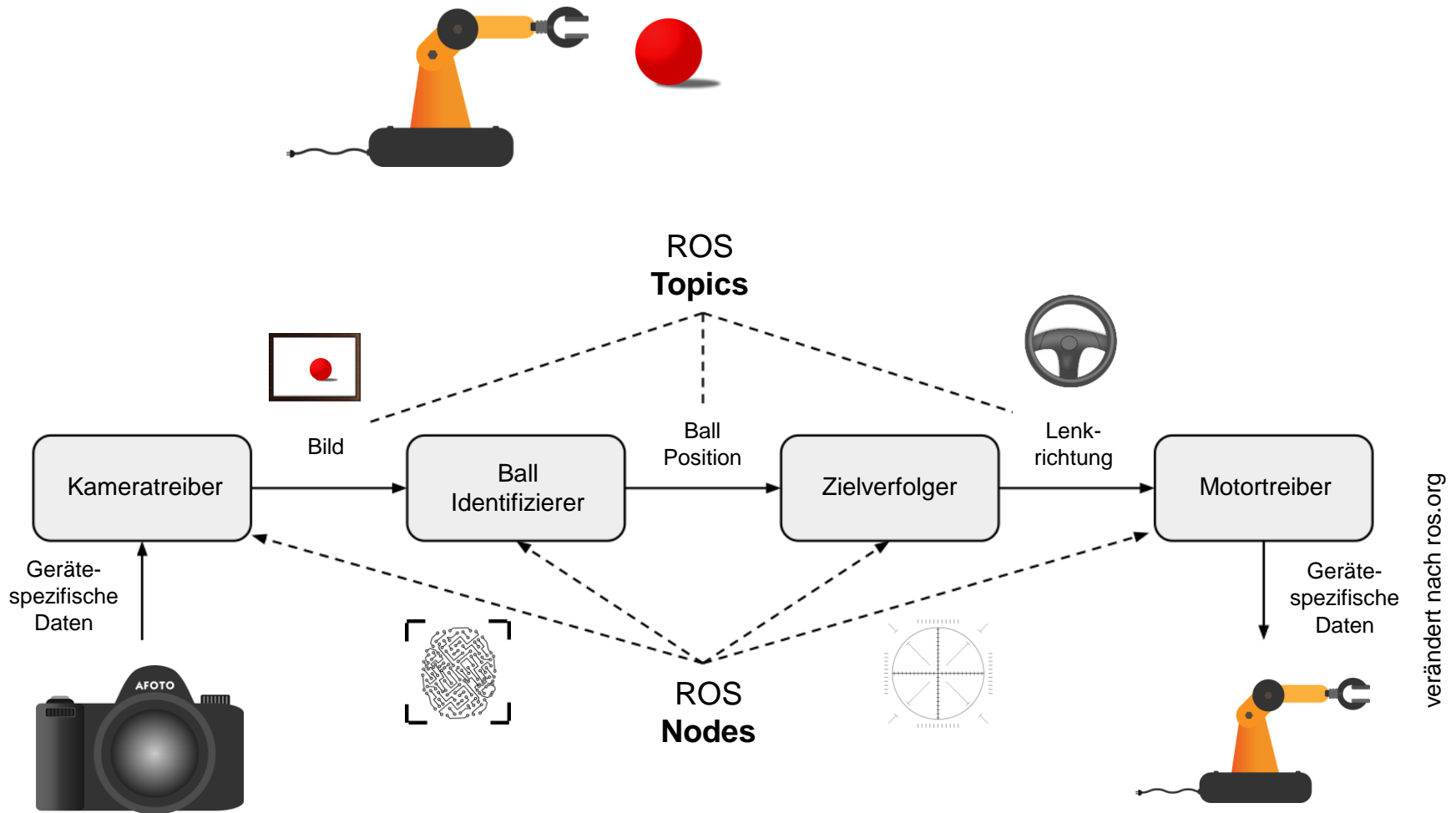
Wintersemester 2022/2023

Dr.-Ing. Sören Vogel

- Open-Source-Framework für vornehmlich Linux-Systeme sowie Programmier- und Skriptsprachen (Python, C++, Matlab)
- In 2007 vom Stanford Artificial Intelligence Laboratory ins Leben gerufen
- Vordefinierte Softwarearchitektur (Schnittstellen, Datentypen, Protokolle, Installationspakete, etc.) für herstellerunabhängige Anwendung
- Klar definierte Schnittstellen zur Interprozesskommunikation
- Große Community mit vielen Beispielen
- Eigentlicher Anwendungszweck zur Entwicklung von (autonomen) Robotern



mandang.com

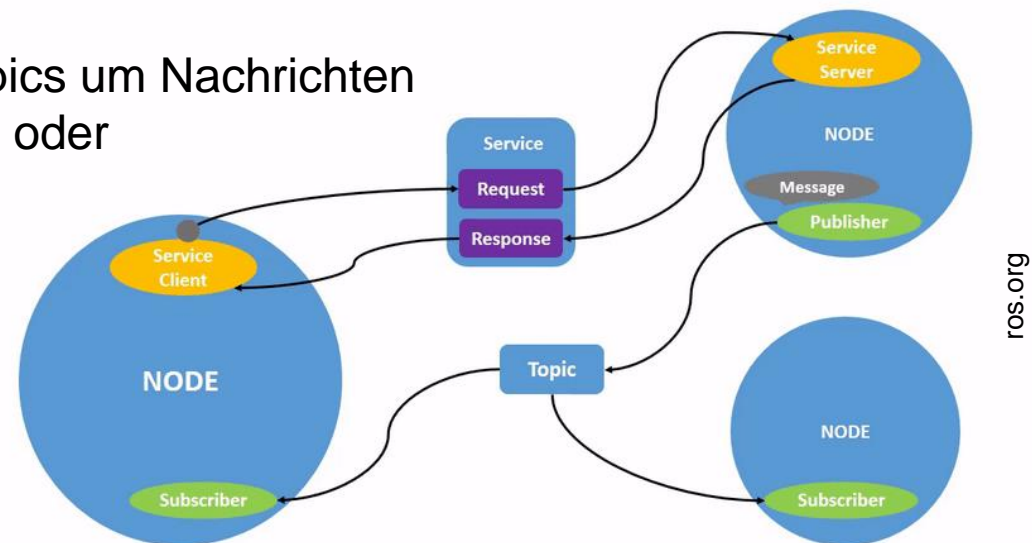


Knoten (Nodes):

- ROS-fähige Anwendungen/Programme (z.B. Sensortreiber)
- Für Verarbeitung von Hardware oder Algorithmen zuständig, wobei jeder Knoten eine eigene Aufgabe hat
- Erlauben sehr modularen Aufbau je nach Anwendungsfall
- Interprozesskommunikation durch den Austausch von Messages zwischen Nodes anhand von
 - **Topics**, indem Nodes in einem Topic **Messages veröffentlichen** (publishen), welche andere Nodes entsprechen **abonnieren** (subscribe) können
 - **Services**, welche Client/Server-Modell entsprechen, wo ein Node (Server) einen Service im System registriert. Andere Nodes können Service **anfragen** (request) und **erhalten Rückmeldung** (response)
- Mehrere Nodes werden in einem Package (Höchste Verwaltungseinheit im ROS) für die Durchführung einer bestimmten Aufgabe zusammengefasst. Mehrere Packages und dessen Zusammenspiel realisieren einen produktiven Prozess.
- Beispiele: Steuerung von Laserscannern, Radmotoren, Algorithmen

Themen (Topics):

- Realisieren einen „Datenstrom von Messages“ als Schnittstelle zwischen einzelnen Anwendungen/Nodes und stellen Ergebnisse der Packages für Datenaustausch zur Verfügung
- Werden verwendet um häufige Messages eines Typs zu senden
- Jedes Topic hat einen eindeutigen Namen und einen definierten Message-Typ
- Nodes verbinden sich mit Topics um Nachrichten zu veröffentlichen (publishen) oder zu abonnieren (subscriben)

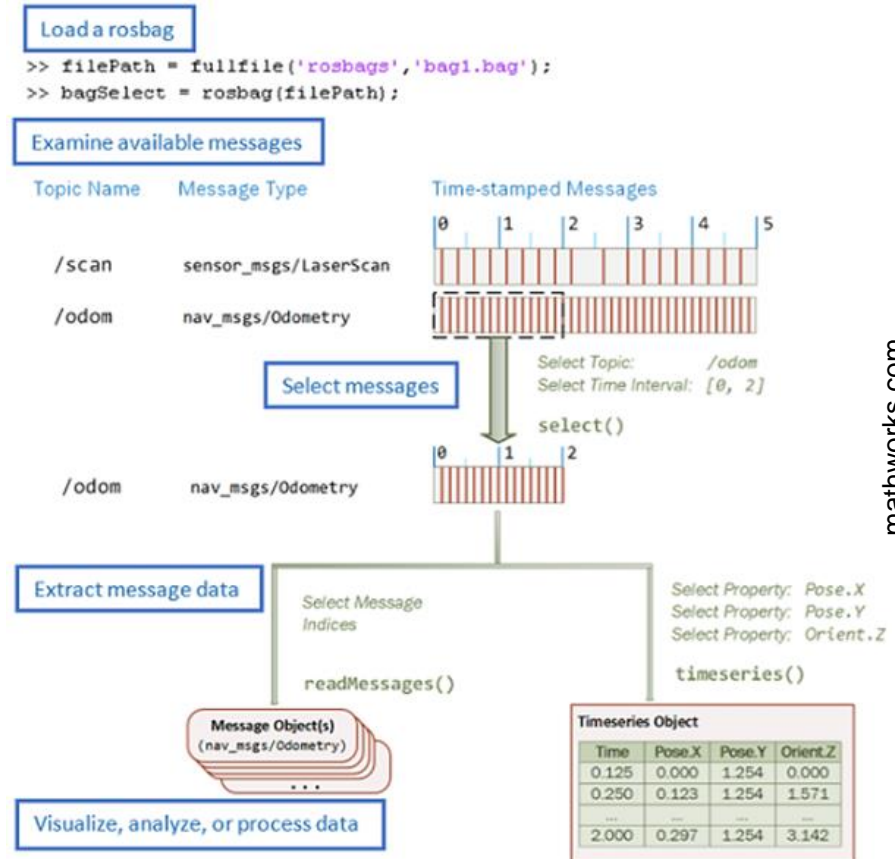


ROS-Bags:

- Aufzeichnung/Archiv zum Abspielen von allen Messages (mit Zeitstempel), welche in einem abonnierten Topic eines Nodes enthalten sind

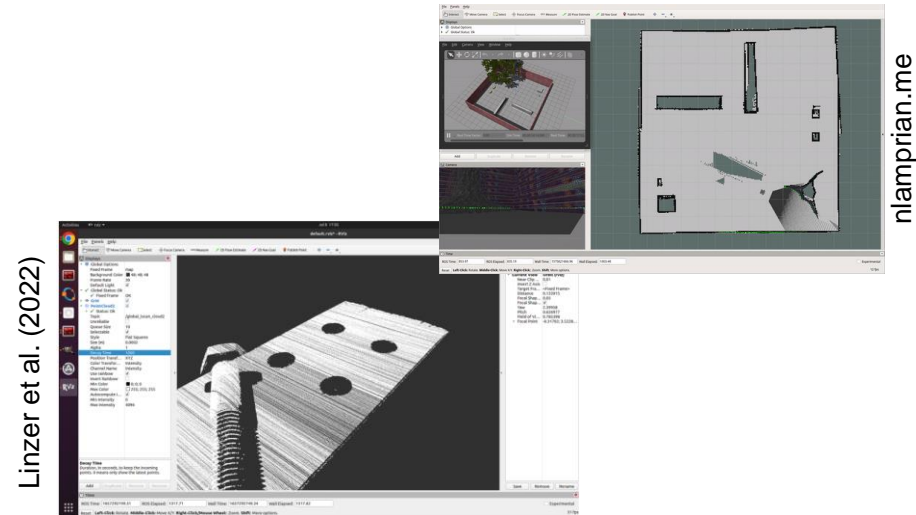
Messages:

- Werden zwischen Nodes übermittelt und beinhalten einen vordefinierten Datentyp zur einheitlichen Formatierung von Daten (Hersteller- bzw. Sensorunabhängig)



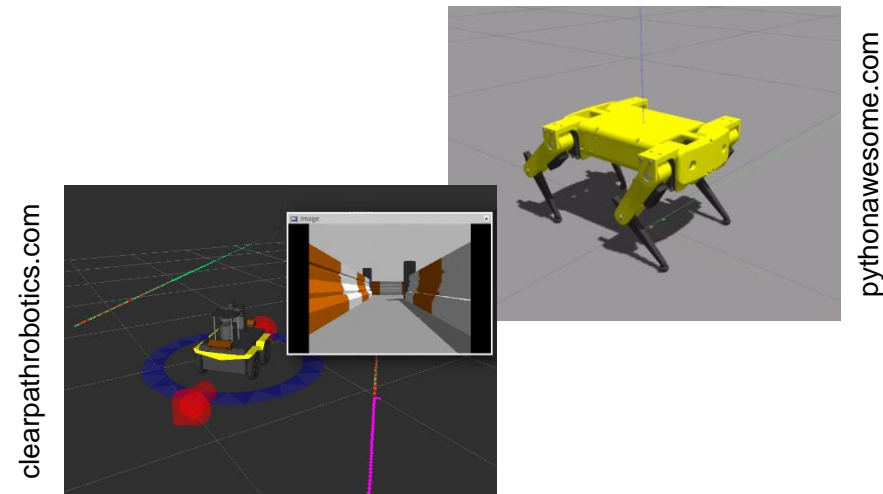
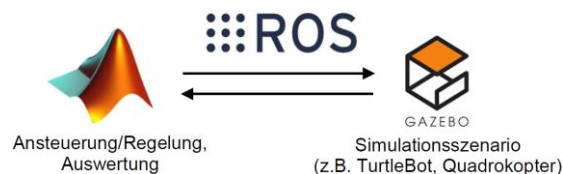
RVIZ:

- Gemeinsame Visualisierung von Sensordaten bzw generell graphische Representation von Messages



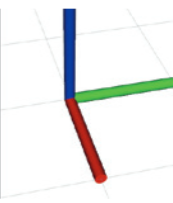
Gazebo:

- Visulationssimulator in einer synthetischen Umgebung
- Simuliert 3D-Starrkörperdynamik und Sensorik (inkl. Messrauschen)



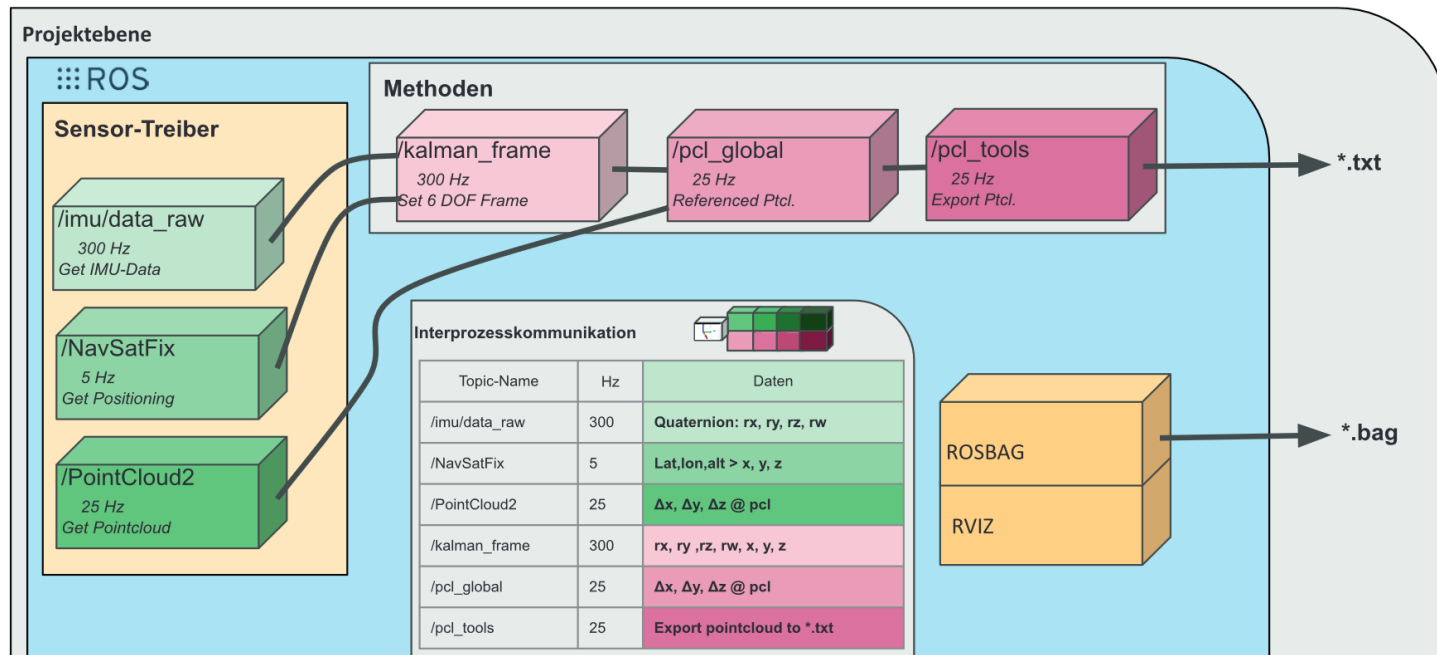
- Herausforderungen bei der Entwicklung von Multi-Sensor-Systemen
 - Neben Messtechnik auch vertieftes Verständnis über Informatik, Mechanik, Elektrotechnik, etc. notwendig
 - Echtzeitsysteme, (An-)Steuerung, Kommunikation, Synchronisation, Datenhaltung, etc.
- ROS für strukturierte Abbildung des komplexen Aufbaus eines MSS nutzen
 - Aufgaben wie Datenerfassung und –analyse werden in einzelne Teilaufgaben strukturiert
- Originäre Messdaten eines Sensor werden mit Hilfe von Treibern innerhalb eines Packages erfasst und in geeignete ROS-Datenstruktur überführt
- Immer mehr Sensorhersteller unterstützen (An-)Steuerung über ROS
- Zugriff auf veröffentlichte Funktionsbibliotheken für vereinfachte Eigenentwicklungen

- Quasi-Standard um einheitlichen Datentyp für gleiche Beobachtungsgrößen unterschiedlicher Hersteller zu haben
- Beschreiben (Mess-)Ergebnis des Sensors/Algorithmus

Datentyp	Beobachtungsgrößen	Frame
geometry_msgs/Transform Message std_msgs/Header header string child_frame_id geometry_msgs/Vector3 translation float64 x float64 y float64 z geometry_msgs/Quaternion rotation float64 x float64 y float64 z float64 w	imu-Frame to velodyne-Frame At time 1537636233.373 – Translation: [0.043, 0.017, -0.085] – Rotation: [0.000, 0.707, 0.000, 0.707]	

Datentyp	Beobachtungsparameter	Abbildung des Messsystems
sensor_msgs/Imu Message std_msgs/Header header geometry_msgs/Quaternion orientation float64[9] orientation_covariance geometry_msgs/Vector3 angular_velocity float64[9] angular_velocity_covariance geometry_msgs/Vector3 linear_acceleration float64[9] linear_acceleration_covariance	header: [...] orientation: x: -0.0065678485147 y: -0.0207788581896 z: 0.112456790974 w: 0.993417622444 [...]	
geometry_msgs/PointStamped std_msgs/Header header geometry_msgs/Point point float64 x float64 y float64 z	header: [...] point: x: 1.93399705402 y: -0.343530434899 z: -0.913549388505	
sensor_msgs/PointCloud2 std_msgs/Header header uint32 height uint32 width sensor_msgs/PointField[] fields [...]	header: [...] height: 1 width: 16126 fields:[...] [...] data:[...]	
sensor_msgs/Twist std_msgs/Header header geometry_msgs/Vector3 linear float64 x float64 y float64 z geometry_msgs/Vector3 angular [...]	header: [...] linear: x: 0.69483762 y: 0.00000000 z: 0.00000000 angular: [...] [...]	
sensor_msgs/Image std_msgs/Header header uint32 height uint32 width string encoding [...] uint8[] data	header: [...] height: 1288 width: 964 encoding:[...] [...] data:[...]	

- Modularer Aufbau von Nodes und entsprechenden Topics zur Referenzierung von Punktwolken mittels GNSS/IMU



Auswahl an ROS-fähigen Systemen

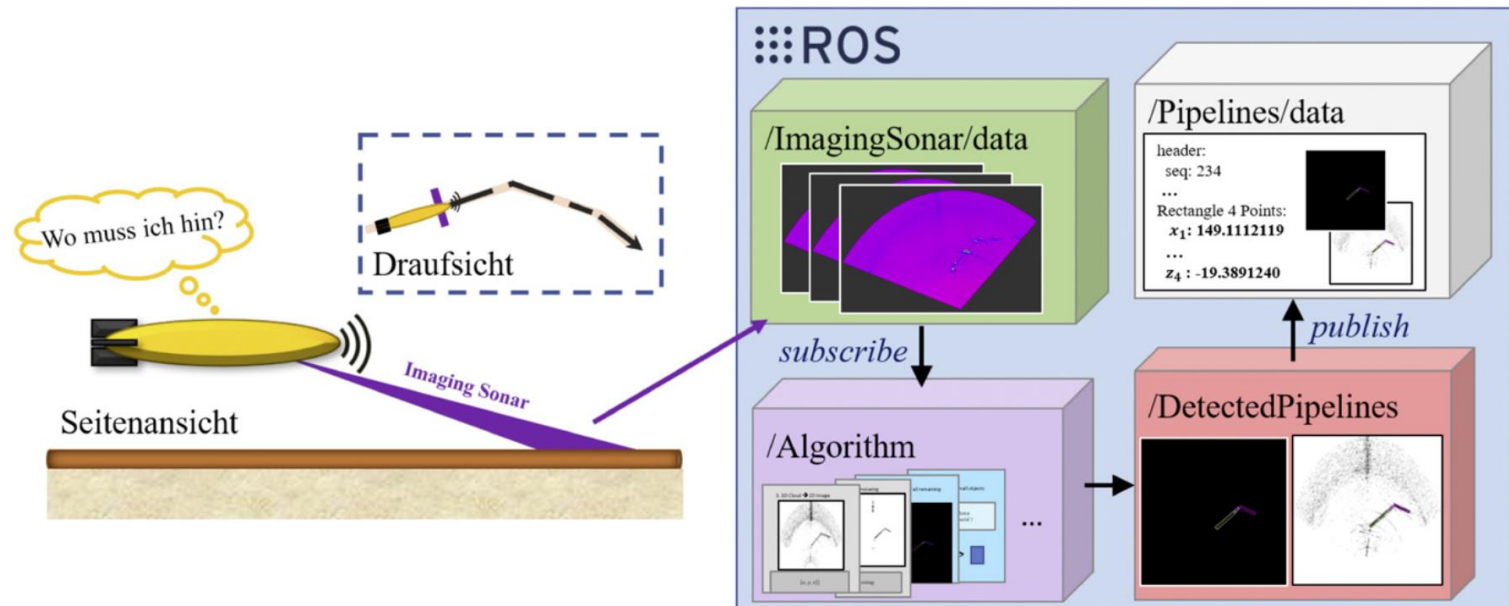


ROS-Packages:

- **husky & husky_robot**: entwickelt von Clearpath, zum Ansteuern des Husky-Roboters, u. a. zum Auslesen der Odometrie
- **leica_lmf_ros & igros_emscon**: entwickelt von der TU Wien, zur Ansteuerung vom Leica AT960- oder LTD800-Lasertracker + T-Scan
- **ros-riegl-vz**: entwickelt von der Firma RIEGL, zur Ansteuerung eines RIEGL VZ-400i 3D-Laserscanners
- **rosig_tps_geocom**: gemeinsame Entwicklung TU Clausthal, LU Hannover, HCU und TU Wien zur Ansteuerung von Leica Tachymetern über GeoCOM
- **septentrio_gnss_driver**: entwickelt von Septentrio, zur GNSS/IMU-gestützten Positionierung eines Systems im Außenbereich
- **sick_scan**: entwickelt von SICK, zur Ansteuerung eines 2D-Laserscanners
- **Universal_Robots_ROS_Driver**: Entwickelt von Universal Robots, zur Ansteuerung des UR5-Roboterarms

- Anwendungsbeispiel: ROS-Node zur Erkennung von **Pipelines** in bildgebenden Sonardaten

数据通道



- Anwendungsbeispiel: ROS-Node zur Erkennung von Pipelines in bildgebenden Sonardaten

