Architectures Logicielles pour la Robotique

Halim Djerroud

LISV
Laboratoire d'ingénierie
des systèmes de Versailles

UVSQ
université PARIS-SACLAY

révision : 1.2

Plan de la présentation

Définition et caractéristiques des robots

Définition :

- Un robot est un système mécanique programmable capable d'effectuer des tâches de manière autonome ou semi-autonome
- Il intègre trois composantes principales :
  - **Perception :** Capteurs pour comprendre l'environnement
  - **Décision :** Algorithmes pour planifier et prendre des décisions
  - **Action :** Actionneurs pour interagir avec l'environnement

Caractéristiques clés des robots :

- Autonomie dans les actions
- Programmabilité pour accomplir diverses tâches
- Interaction avec l'environnement physique

Types de robots et applications

Classification des robots :

- **Robots industriels :**
  - Automatisation des tâches en usine (assemblage, soudage).
- **Robots mobiles :**
  - Déplacement autonome (drones, véhicules autonomes).
- **Robots humanoïdes :**
  - Interaction sociale, assistance (robots domestiques, compagnons).
- **Robots médicaux :**
  - Chirurgie assistée, rééducation.
- **Robots spécialisés :**
  - Exploration (robots sous-marins, martiens).

Applications :

- Automatisation industrielle.
- Transport autonome.
- Exploration scientifique.
- Assistance à la personne.

Types de Robots

Catégorie	Sous-types	Exemples
Navigation	AMR, Drones, Véhicules autonomes	Roomba, Tesla, Wing
Exploration	Robots planétaires, sous-marins	Curiosity, ROV, Fukushima bots
Collaboration	Cobots, Robots médicaux, Essaims robotiques	Da Vinci, Universal Robots, Buddy
Spécialisés	Militaires, Scientifiques, Artistiques	Drones militaires, Boston Dynamics
Divertissement	Jouets, Robots interactifs	Cozmo, LEGO Mindstorms

Environnements robotiques : Définition et catégories

Définition :

- L'environnement robotique correspond au contexte physique ou virtuel dans lequel un robot évolue.
- Il influence directement les capteurs, les algorithmes et le comportement du robot.

Catégories d'environnements :

- **Structuré :**
  - Organisation claire et prévisible (ex. usines, entrepôts).
  - Permet une planification précise.
- **Semi-structuré :**
  - Partiellement organisé avec des éléments imprévus (ex. bâtiments publics).
  - Nécessite une réactivité accrue.
- **Non structuré :**
  - Aucun ordre ni organisation prévisible (ex. forêts, terrains inconnus).
  - Implique des algorithmes d'adaptation en temps réel.

Comparaison des types d'environnements

Caractéristiques :

Type d'environnement	Caractéristiques	Exemples
Structuré	Prévisible, organisé	Usines, entrepôts
Semi-structuré	Partiellement organisé	Bâtiments, zones urbaines
Non structuré	Chaotique, imprévisible	Forêts, terrains inconnus

Implications pour les robots :

- Plus l'environnement est structuré, plus les algorithmes peuvent être délibératifs.
- Les environnements non structurés nécessitent des approches réactives et adaptatives.

Qu'est-ce qu'une architecture logicielle robotique ?

- Organisation et structure des systèmes logiciels pour robots
- Interaction entre capteurs, actionneurs et algorithmes
- Objectifs principaux :
  - Modularité
  - Scalabilité
  - Robustesse

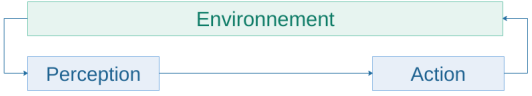
Approches principales

- **Centralisée (Cours 1):**
  - Tout le traitement géré par un système central
  - Simple mais peu scalable
- **Distribuée : (Cours 2 - SMA)**
  - Sous-systèmes autonomes
  - Meilleure robustesse, complexité accrue
- **Hybride :**
  - Combinaison des deux approches précédentes

Type d'architecture robotique

Architecture hiérarchique

Architecture réactive



Architecture hybride

Composants d'une architecture robotique

Matériel :

- Capteurs (caméras, LIDAR, etc.)
- Actionneurs (moteurs, pinces, bras, ...)

Logiciel :

- Perception (SLAM, vision)
- Planification et contrôle
- Communication

Les briques logicielles

1. **Perception :** Acquisition et traitement des données des capteurs
2. **Modélisation :** Représentation interne de l'environnement et de l'état du robot
3. **Planification :** Génération de plans d'action (global, local, temporel)
4. **Prise de décision :** Sélection des actions en fonction des objectifs et des événements
5. **Contrôle :** Exécution des actions avec précision et robustesse
6. **Communication :** Échange de données entre modules et systèmes externes
7. **Interface utilisateur :** Interaction et surveillance par l'humain

1. Perception

Objectif : Acquérir et traiter les données des capteurs pour interpréter l'environnement.

- **Acquisition de données :**
  - LIDAR, caméras, capteurs de proximité, IMU
- **Traitement :**
  - Filtrage (ex. filtre de Kalman)
  - Extraction de caractéristiques (ex. détection d'objets)
- **Représentation :**
  - Cartographie (SLAM)
  - Détection et suivi des obstacles

2. Modélisation

Objectif : Représenter l'état interne et l'environnement externe du robot.

- **Modélisation de l'état du robot :**
  - Position, orientation.
  - Modèles cinématiques et dynamiques.
- **Modélisation de l'environnement :**
  - Cartes 2D/3D (grilles d'occupation, nuages de points)
  - Modèles probabilistes.
- **Modèles prédictifs :**
  - Prédiction des mouvements des obstacles.

3. Planification

Objectif : Définir les actions nécessaires pour atteindre un objectif

- **Planification globale :**
  - Recherche de chemin (A*, D* Lite, Dijkstra, etc.)
- **Planification locale :**
  - Évitement des obstacles dynamiques
- **Planification des tâches :**
  - Décomposition des objectifs complexes
- **Planification temporelle :**
  - Coordination des actions avec contrainte de temps

4. Prise de décision

Objectif : Traduire les plans en actions concrètes et s'adapter aux changements

- **Règles comportementales :**
  - Réactions aux événements imprévus
- **Stratégies de haut niveau :**
  - Automates d'état, arbres de décision
- **Algorithmes d'intelligence artificielle :**
  - Apprentissage par renforcement.
  - Modèles prédictifs

5. Contrôle

Objectif : Exécuter les commandes avec précision et stabilité.

- **Contrôle bas niveau :**
  - Commande des actionneurs (moteurs, pinces)
- **Contrôle cinématique :**
  - Calcul des trajectoires (bras robotique, roues)
- **Contrôle dynamique :**
  - Gestion des forces et couples
- **Contrôle robuste/adaptatif :**
  - Gestion des incertitudes (ex. surfaces glissantes)

6. Communication

Objectif : Permettre l'échange de données entre composants ou systèmes externes

- **Middleware :**
  - ROS, DDS, YARP
- **Protocoles de communication :**
  - CAN, MQTT, Ethernet industriel
- **Systèmes distribués :**
  - Coordination multi-robots

7. Interface utilisateur

Objectif : Contrôler, surveiller et interagir avec le robot

- **Interfaces graphiques :**
  - Dashboards pour visualisation en temps réel
- **Commande à distance :**
  - Téléopération (joystick, smartphone)
- **Interfaces naturelles :**
  - Commandes vocales ou gestuelles

Architecture hiérarchique

Caractéristiques principales :

- Organisation en couches hiérarchiques
- Chaque couche a un rôle spécifique (planification, contrôle, exécution)
- Communication descendante et ascendante :
  - Les couches supérieures supervisent les couches inférieures
  - Les couches inférieures fournissent des retours d'information

Avantages :

- Structure modulaire et bien organisée
- Facile à décomposer en sous-systèmes
- Appropriée pour des tâches complexes nécessitant de la planification

Inconvénients :

- Réponse lente dans des environnements dynamiques
- Sensible aux erreurs dans les couches supérieures

Architecture réactives

Caractéristiques principales :

- Absence de planification globale
- Basée sur des comportements spécialisés (éviter obstacles, suivre cible, etc.)
- Réactivité et rapidité en temps réel
- Exécution parallèle des comportements avec mécanisme d'arbitrage

Avantages :

- Simplicité et robustesse
- Adaptée aux environnements dynamiques

Inconvénients :

- Pas de planification à long terme
- Comportements parfois imprévisibles
- Limitation des tâches complexes

Exemple : Architecture Subsomption (Rodney Brooks)

Architectures hybrides

Caractéristiques principales :

- Combine les architectures réactives et délibératives
- Trois niveaux principaux :
  - **Planification :** prise de décision à long terme
  - **Coordination :** traduction des plans en actions concrètes
  - **Réactivité :** exécution rapide des actions en temps réel
- Utilise une hiérarchie modulaire pour garantir flexibilité et robustesse

Avantages :

- Combine la réactivité rapide avec la planification stratégique
- Appropriée pour des tâches complexes dans des environnements dynamiques

Inconvénients :

- Complexité de conception et d'implémentation
- Coordination entre niveaux parfois délicate

Middleware robotique

Exemples populaires :

- **ROS (Robot Operating System) :**
  - Bibliothèques, outils pour le développement
  - Gestion de la communication
- **YARP :**
  - Spécialisé pour les robots humanoïdes
- **MOOS :**
  - Dédié aux systèmes marins

Défis de l'architecture logicielle robotique

- Gestion des temps réels
- Interopérabilité entre composants divers
- Évolutivité pour de nouveaux capteurs/actionneurs
- Robustesse face aux pannes
- Sécurité des données et des systèmes

L'architecture logicielle

- L'architecture logicielle est clé pour le fonctionnement des robots
- Nécessite une approche multidisciplinaire :
  - Intelligence artificielle
  - Informatique
  - Systèmes embarqués
- ROS reste une solution de référence dans le domaine

Exemple d'architecture : L'architecture LAAS

Caractéristiques principales :

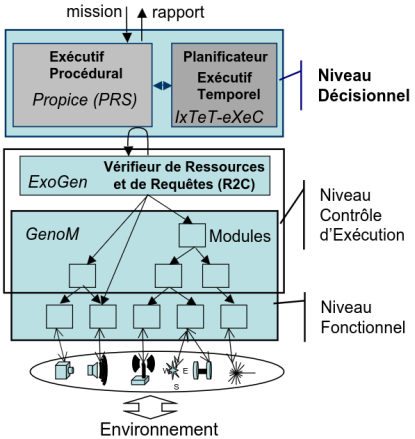
- Architecture hiérarchique à trois niveaux :
  - **Niveau décisionnel :** Planification des tâches à long terme (ex. mission planning)
  - **Niveau exécutif :** Supervise et coordonne les tâches planifiées
  - **Niveau fonctionnel :** Exécution directe des commandes via capteurs et actionneurs
- Basée sur des boucles de rétroaction pour ajuster les actions en fonction des retours d'état

Avantages :

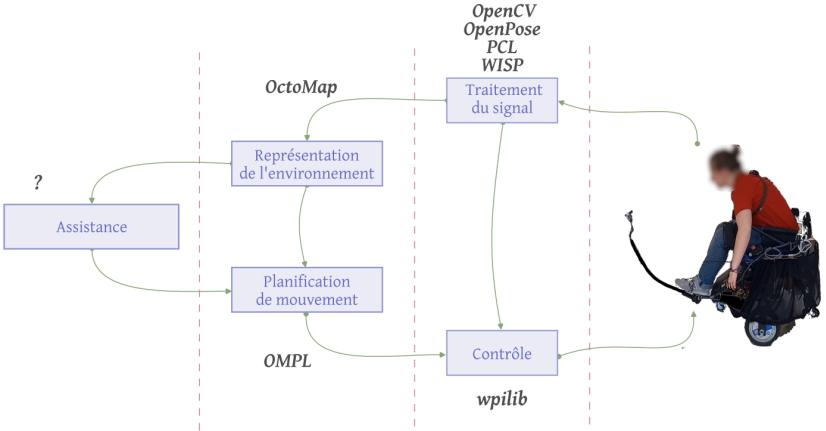
- Séparation claire des responsabilités entre les niveaux
- Flexibilité pour intégrer des tâches complexes.

Applications :

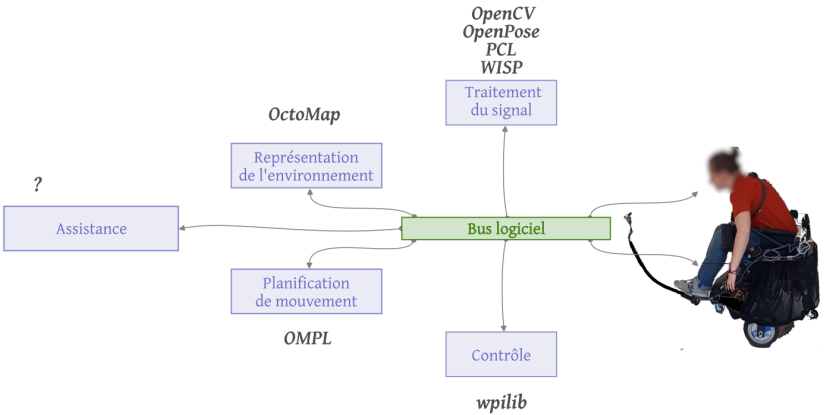
- Utilisée dans des systèmes robotiques complexes, comme



Structure logiciel d'un robot



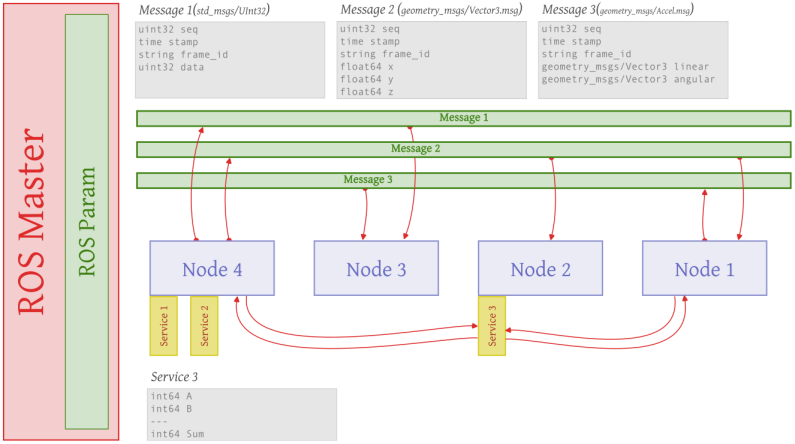
Structure logiciel d'un robot en utilisant ROS



ROS : C'est quoi ?

- 1 - Une infrastructure
  - Messages
  - Services
  - Paramètres
  - etc.
- 2 - Des outils
  - Débogage
  - Visualisation
  - Compilation
  - etc.
- 3 - Des bibliothèques intégrées utiles
  - Traitement de signal
  - Planification
  - Contrôle
  - etc.
- 4 - Un écosystème
  - Une communauté libre
  - Une industrie impliquée
  - Des plateformes robotiques
  - etc.

Structure logiciel d'un robot en utilisant ROS



Architecture de communication inter-processus

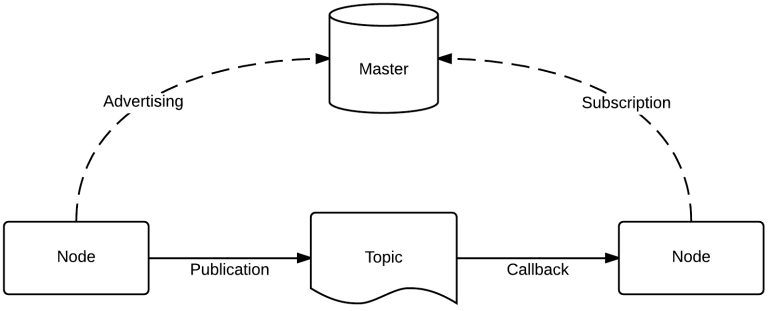


Figure: https://fr.wikipedia.org/wiki/Robot_Operating_System

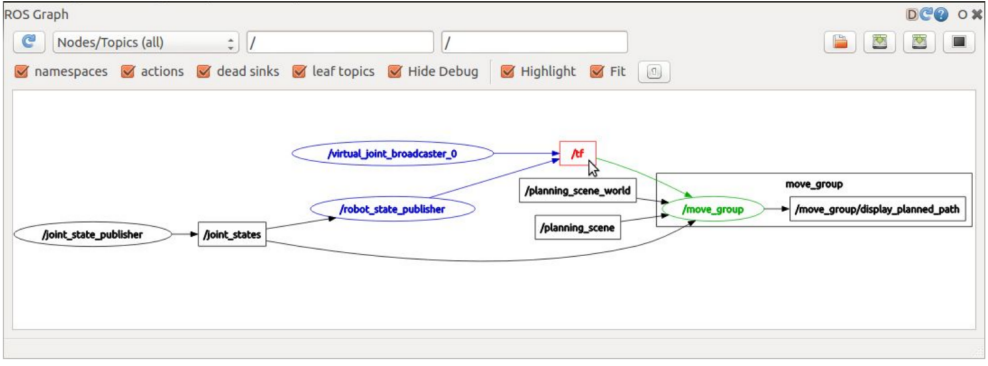
ROS : Un ensemble d'outils

- Des bibliothèques permettant de créer des exécutables
- Un ensemble de messages standard et des outils pour créer de nouveaux messages
- Un outil de compilation *catkin* (basé sur CMake)
- Impose une organisation standardisée du projet de développement
- Architecture de communication inter-processus ou via le protocole TCP
- Un serveur de paramètres
- Un ensemble d'outils de débogage

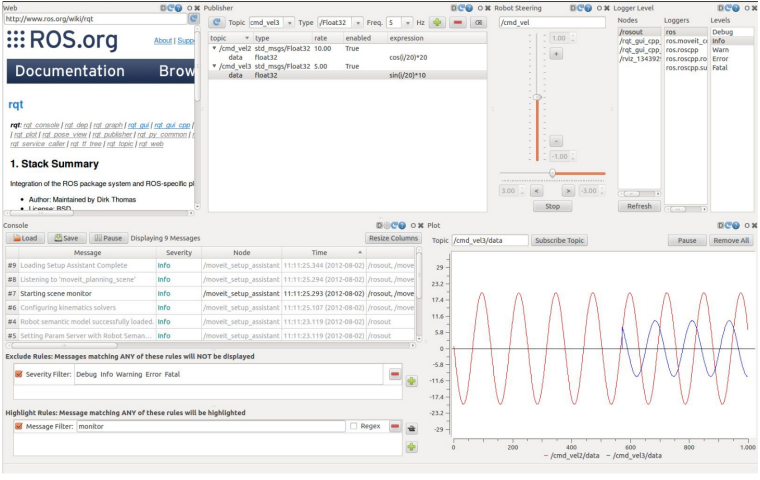
Quleques outils

- Catkin
- Des commandes pour tout faire
- rosbag
- rqt (ex : rqt_graph)
- RViz
- et bien d'autres ...

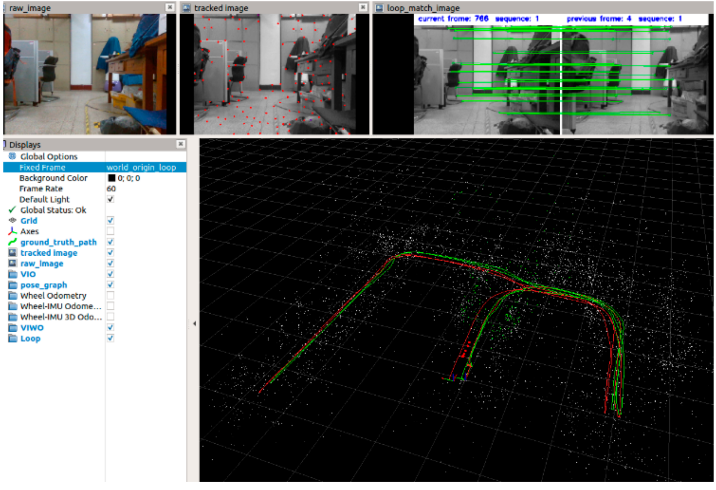
rqt_graph


ROS Graph
Nodes/Topics (all)
/
/
namespaces
actions
dead sinks
leaf topics
Hide Debug
Highlight
Fit
/virtual_joint_broadcaster_0
/tf
/planning_scene_world
move_group
/robot_state_publisher
/move_group
/move_group/display_planned_path
/joint_state_publisher
/joint_states
/planning_scene

rqt


Web
http://www.ros.org/wiki/rqt
ROS.org
About | Supp
Documentation
Brow
rqt
rqt: rqt_console | rqt_dep | rqt_graph | rqt_gui | rqt_gui_cpp |
rqt_plot | rqt_pose_view | rqt_publisher | rqt_py_common |
rqt_service_caller | rqt_tf_tree | rqt_topic | rqt_web
1. Stack Summary
Integration of the ROS package system and ROS-specific pl
Author: Maintained by Dirk Thomas
Publisher
Topic: cmd_vel3 Type: /Float32 Freq: 5 Hz
topic | type | rate | enabled | expression
/cmd_vel2 std_msgs/Float32 10.00 True
data float32 cos(j/20)*20
/cmd_vel3 std_msgs/Float32 5.00 True
data float32 sin(j/20)*10
Robot Steering
/cmd_vel
1.00
-1.00
3.00
-3.00
Stop
Logger Level
Nodes
/rosout
/rqt_gui_cpp
/rqt_gui_cpp
/rviz_134392
Loggers
/ros
ros.moveit_c
ros.roscpp
ros.roscpp.ro
ros.roscpp.su
Levels
Debug
Info
Warn
Error
Fatal
Refresh
Console
Load
Save
Pause
Displaying 9 Messages
Resize Columns
Message | Severity | Node | Time
#9 Loading Setup Assistant Complete | Info | /moveit_setup_assistant | 11:11:25.344 (2012-08-02) | /rosout, /move
#8 Listening to 'moveit_planning_scene' | Info | /moveit_setup_assistant | 11:11:25.294 (2012-08-02) | /rosout, /move
#7 Starting scene monitor | Info | /moveit_setup_assistant | 11:11:25.293 (2012-08-02) | /rosout, /move
#6 Configuring kinematics solvers | Info | /moveit_setup_assistant | 11:11:25.107 (2012-08-02) | /rosout, /move
#4 Robot semantic model successfully loaded. | Info | /moveit_setup_assistant | 11:11:23.119 (2012-08-02) | /rosout
#5 Setting Param Server with Robot Seman... | Info | /moveit_setup_assistant | 11:11:23.119 (2012-08-02) | /rosout
Exclude Rules: Messages matching ANY of these rules will NOT be displayed
Severity Filter: Debug Info Warning Error Fatal
Highlight Rules: Message matching ANY of these rules will be highlighted
Message Filter: monitor
Regex
Plot
Topic: /cmd_vel3/data
Subscribe Topic
Pause
Remove All
29
23.2
17.4
11.6
5.8
0
-5.8
-11.6
-17.4
-23.2
-29
0
200
400
600
800
1,000
/cmd_vel2/data
/cmd_vel3/data

RViz

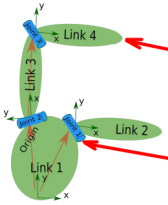

raw_image
tracked image
loop_match_image
current frame: 766 sequence: 1
previous frame: 4 sequence: 1
Displays
Global Options
Fixed Frame
world_origin_loop
Background Color
0; 0; 0
Frame Rate
60
Default Light
Global Status: Ok
Grid
Axes
ground_truth_path
tracked image
raw_image
VIO
pose_graph
Wheel Odometry
Wheel-IMU Odome...
Wheel-IMU 3D Odo...
VIWO
Loop

Les fonctionnalités

- Des fonctionnalités implémentées par des bibliothèques spécialisées
- Issues de plusieurs années de recherche
- Parmi les plus populaires :
  - Contrôle
  - Navigation (2D / 3D)
  - Planification de mouvement (bras robotisés)
  - Planification de missions
  - Vision
  - etc.

Modèle du robot : fichier URDF

URDF : Unified Robot Description Format



robot.urdf

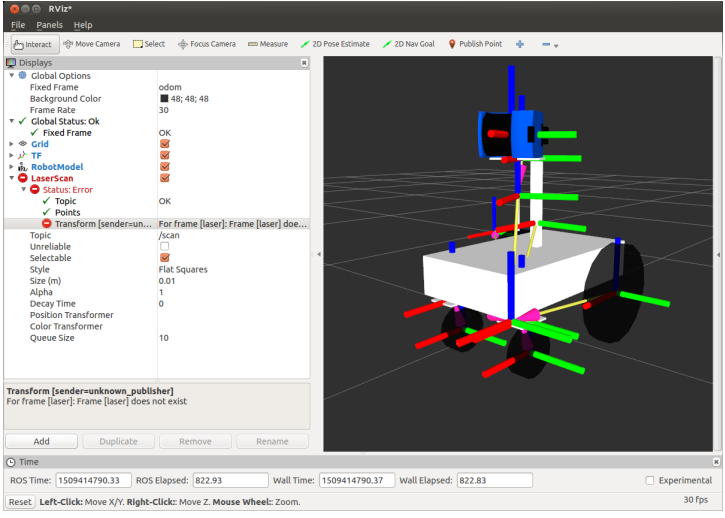
```
<robot name="robot">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint> .... </joint>
  <joint> .... </joint>
  <joint> .... </joint>
</robot>
```

```
<link name="link_name">
  <visual>
    <geometry>
      <mesh filename="mesh.dae"/>
    </geometry>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" .../>
  </inertial>
</link>
```

```
<joint name="joint_name" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" upper="0.548" ... />
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="parent_link_name"/>
  <child link="child_link_name"/>
</joint>
```

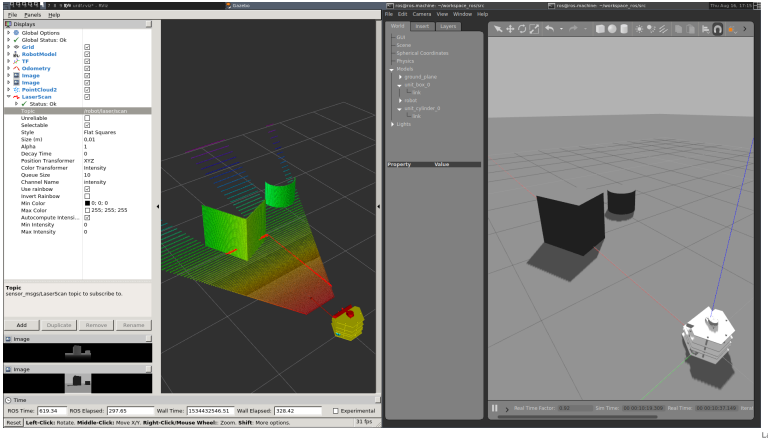
LISV Laboratoire d'Ingénierie des systèmes de Versailles — UVSQ université PARIS-SACLAY

ROS TF

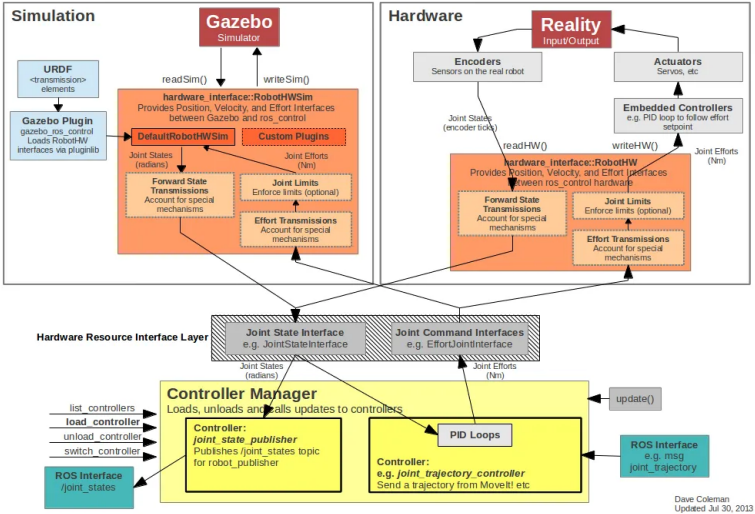

RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Global Options
Fixed Frame odom
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
TF
RobotModel
LaserScan
Status: Error
Topic OK
Points
Transform [sender=un... For frame [laser]: Frame [laser] doe...
Topic /scan
Unreliable
Selectable
Style Flat Squares
Size (m) 0.01
Alpha 1
Decay Time 0
Position Transformer
Color Transformer
Queue Size 10
Transform [sender=unknown_publisher]
For frame [laser]: Frame [laser] does not exist
Add Duplicate Remove Rename
Time
ROS Time: 1509414790.33 ROS Elapsed: 822.93 Wall Time: 1509414790.37 Wall Elapsed: 822.83
Experimental
Reset Left-Click: Move X/Y. Right-Click:: Move Z. Mouse Wheel:: Zoom.
30 fps

Simulation avec Gazebo

SDF : Simulation Description Format

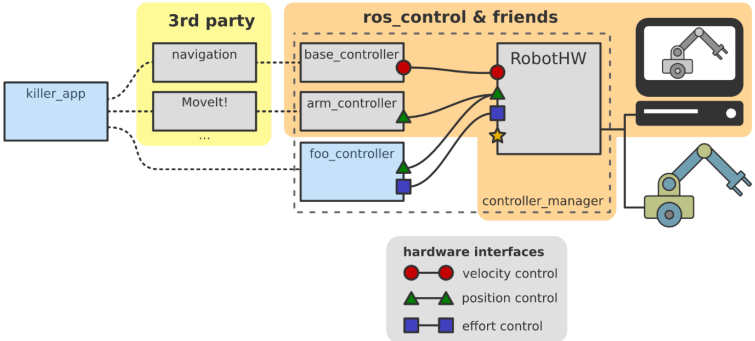


ROS Control

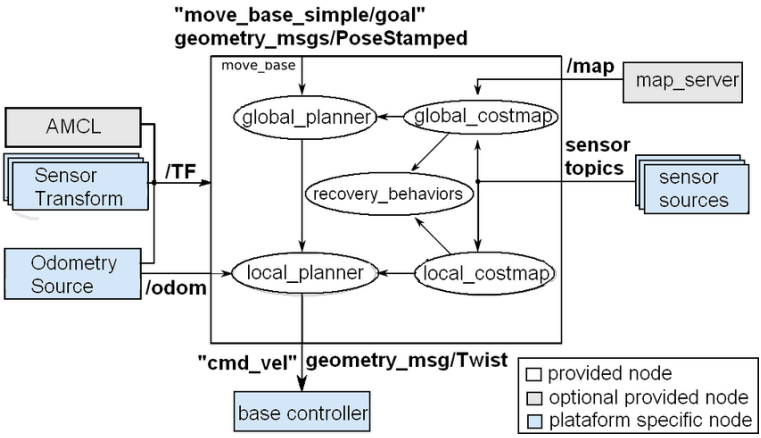


Dave Coleman
Updated Jul 30, 2013

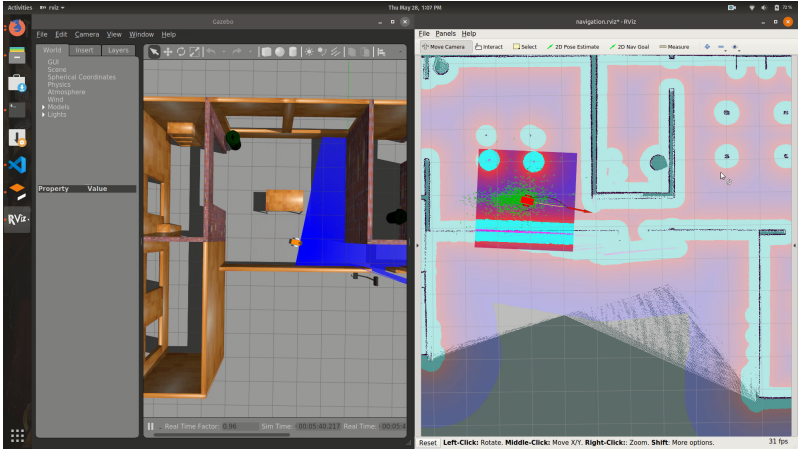
ROS Control (suite)



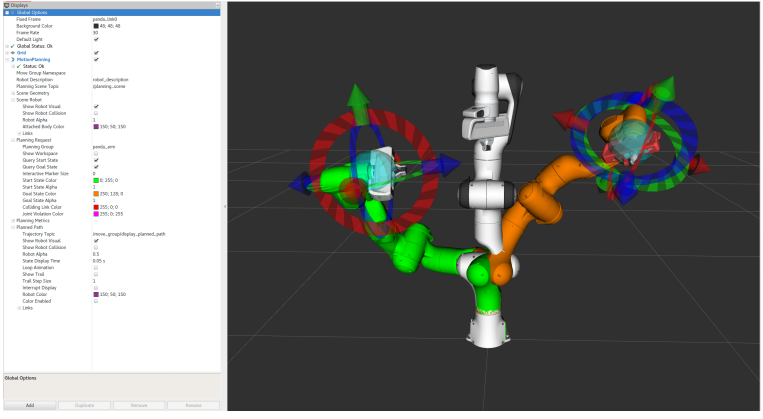
La pile de navigation (navigation stack)



La pile de navigation (navigation stack)



MoveIt


Displays
Global Options
Fixed Frame
panda_link0
Background Color
48; 48; 48
Frame Rate
30
Default Light
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
/planning_scene
Scene Geometry
Scene Robot
Show Robot Visual
Show Robot Collision
Robot Alpha
1
Attached Body Color
150; 50; 150
Links
Planning Request
Planning Group
panda_arm
Show Workspace
Query Start State
Query Goal State
Interactive Marker Size
Start State Color
0; 255; 0
Start State Alpha
1
Goal State Color
250; 128; 0
Goal State Alpha
1
Colliding Link Color
255; 0; 0
Joint Violation Color
255; 0; 255
Planning Metrics
Planned Path
Trajectory Topic
/move_group/display_planned_path
Show Robot Visual
Show Robot Collision
Robot Alpha
0.5
State Display Time
0.05 s
Loop Animation
Show Trail
Trail Step Size
1
Interrupt Display
Robot Color
150; 50; 150
Color Enabled
Links
Global Options
Add
Duplicate
Remove
Rename

Communauté

- Willow Garage (2007-2013)
- Soutenu par plusieurs industriels
- OSRF => OpenRobotics (ROS, Gazebo, Open-RMF)
- Soutenu par : ROS Industrial Consortium
- Les constructeurs de capteurs et d'actionneurs fournissent des modules ROS pour leur matériel
- Documentation et tutoriels
- ROSConf / ROSConFr (pour cette année du 17 au 20 juin 2024 à Nantes)

Qui utilise ROS ?

- Supporté nativement par plus de 80 plateformes robotiques

LISV Laboratoire d'ingénierie des systèmes de Versailles | UVSQ université PARIS-SACLAY

Qui utilise ROS ?

- DARPA Robotics Challenge
- Lunabotics Challenge (NASA)

Intérêt pour la recherche

- **Avant ROS :**
  1. Réaliser une architecture logiciel spécifique pour transporter les données (selon le matériel à disposition)
  2. Intégrer les fonctionnalités de base à travers de bibliothèques (gros travail d'intégration). Ex : localisation, navigation, contrôle ...
  3. Réaliser des tests
  4. --- *Le prototype est prêt pour la recherche* --- 70 à 90 % du temps de projet est écoulé
  5. - de 30 % de temps pour réaliser son travail de recherche
- **Après ROS :**
  1. ROS contient déjà l'architecture logiciel, l'environnement d'exécution, de test, de simulation et de récupération de données
  2. Il suffit de sélectionner et d'intégrer les fonctionnalités de base pour son projet de recherche
  3. --- *Le prototype est prêt pour la recherche* --- 10 à 30 % du temps de projet est écoulé
  4. + de 70 % de temps pour réaliser son travail de recherche

Intérêt pour le chercheur

- Intégrer les avancées réalisées dans un package ROS pour permettre :
  - À d'autres équipes de recherche de reproduire l'expérience plus facilement
  - D'être utilisées par d'autres équipes comme brique de base pour leurs recherches

Conclusion

- ROS se positionne comme un facilitateur de projets robotiques
- Passer moins de temps à créer un nouvel écosystème pour chaque nouveau projet + économies (temps et finances)
- Se concentrer sur la recherche

Perspectives : ROS 2

- Il existe deux versions du middleware ROS : ROS et ROS2
- ROS2 est pleinement fonctionnel (depuis peu de temps)
- La plupart des packages sont déjà publiés et fonctionnels
- Utilise un DDS (Data Distribution Service) => OMG
- Multi-robots (MRS)
- Temps réel
- uRos
- Réseaux instables
- Adapté aux normes industrielles