R5-COP

Reconfigurable ROS-based Resilient Reasoning Robotic Cooperating Systems

Planning and Reasoning Architecture

Tadeusz Dobrowiecki (BME)

Project	R5-COP	**Grant agreement no.**	621447
Deliverable	D26.12	**Date**	01/31/2016
Contact Person	Tadeusz Dobrowiecki	**Organization**	BME
E-Mail	dobrowiecki@mit.bme.hu	**Diss. Level**	PU

Document History			
Ver.	**Date**	**Changes**	**Author**
0.0	7.04.2016	Lay-out of the deliverable	Tadeusz Dobrowiecki (BME)
0.1	10.07.2016	Content of the report	Tadeusz Dobrowiecki (BME)
0.2	20.07.2016	Editing.	Tadeusz Dobrowiecki (BME)
0.3	25.07.2016	Illustrations.	Péter Eredics, István Engedy (BME)
0.4	31.07.2016	Internal review	Jon Azpiazu (SINTEF), Thomas Madsen Almdal (DTI), István Majzik (BME)
0.5	31.07.2016	Final editing.	Tadeusz Dobrowiecki (BME)

Note: Filename should be

“R5-COP_D##_#.doc”, e.g. „R5-COP_D91.1_v0.1_TUBS.doc“

Fields are defined as follow

1. Deliverable number	***.***
2. Revision number:	
draft version	**v**
approved	**a**
version sequence (two digits)	***.***
3. Company identification (Partner acronym)	*****

Content

List of Acronyms

AGP	Art Gallery Problem
AMCL	Adaptive (or KLD-sampling) Monte Carlo Localization
AUV	Autonomous Underwater Vehicle
CFR	Co-worker (and First Responder) Robot
CTSP	Clustered (or Colored) Traveling Salesman Problem
DWA	Dynamic Window Approach
EKF	Extended Kalman Filter
FMP	Functional Mission Plan
HATP	Hierarchical Agent based Task Planner
HTN	Hierarchical Task Network
IMR	Inspection, Maintenance and Repair
JSON	Java Script Object Notation
KDB	Knowledge Databases
KM	Knowledge Management
KML	Keyhole Markup Language
KR	Knowledge Representation
HMI	Human Machine Interface
LKH	Lin-Kernighan Heuristics, Lin-Kernighan-Helsgaun Algorithm
ME	Mixture of Experts
MMP	Metric Mission Plan
OpC	Process Operation Center
PDDL	Planning Domain Definition Language
POCL	Partial Order Causal Link (planner)
PRA	Planning and Reasoning Architecture
ROpC	Remote Operation Center
ROS	Robot Operating System
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle

1 Introduction

1.1 Summary

The final version of the Planning and Reasoning Architecture presents architecture solution for the SINTEF co-worker robot, working in a gas and oil plant as an autonomous remote inspector (R5-COP D11.10). The report introduces the problem, discusses the requirements, and proposes a high level modular architecture, discussing shortly the function of particular modules.

1.2 Purpose of document

The aim of the final version of the Planning and Reasoning Architecture is to show how the general considerations about the role of planning and reasoning in the autonomous robots, discussed in the first version of this deliverable (R5-COP D26.11) can be applied to design an architecture when the research is focused on a particular robotic problem. The chosen pilot problem is analysed for the functional hints and architecture is proposed bridging high level requirements to the low level ROS package capabilities.

The chosen pilot problem is an involved and complex planning problem. It is a single demonstrator case where essential part of the task is not to navigate the robot around and make continuous visual or range observations, but actually to become stationary, to orient itself and to use a wide assortment of non-visual, non-range finding sensory equipment to observe the state of the objects in the environment.

The planning for the stationary non-visual sensing is quite different from the map/marker/waypoint planning of the optimal trajectory in the metric or in the configuration space. Such planning involves hierarchy of (measurement) functions, repetitive measurement actions, dependence of the subsequent measurements on the results of the previous ones, and the option of the frequent consultation with the operator.

The primary challenge in putting together the planning and reasoning architecture for such problem is to assure that plan fragments of different character are integrated seamlessly and can be executed efficiently. Such architecture is proposed, discussed in detail, together with alternative options and yet to solve open problems.

1.3 Partners involved

Partners and Contribution		
Short Name	**Full Name**	**Contribution**
BME	Budapest University of Technology and Economics (HU)	Preparation of the report
SINTEF	Stiftelsen SINTEF (NO)	Internal review
DTI	Danish Technological Institute (DK)	Internal review

2 The application domain - Co-worker robot for industrial inspection

After the general review of the literature on planning in robotics in D26.11, a particular class of robotic systems addressed in the project was chosen to make the architecture design more specific and embedded in a concrete application context (R5-COP D26.11).

Among the possible candidates deployed by the R5-COP demonstrators an industrial co-worker robotic system was chosen, represented by the gas-and-oil inspecting co-worker robot of SINTEF Company (WP41 Co-worker)[1]. The rationale of such choice was that an efficient and effective maintenance in high risk enterprises, as e.g. the oil and gas facilities, possesses many challenging characteristics, like managing extensive production facilities and huge number of control points, using multiple inspection technologies, in hazardous working conditions, all of these making the planning for the co-worker which autonomously executes inspection tasks in extensive industrial plants a challenging problem.

Industrial inspection problems (Inspection, Maintenance and Repair – IMR) may involve various configurations of the inspecting service robots, like:

(1) Ground robots making inspections over dirt roads and an open country site (e.g. inspecting solar power plants (Kim 2011, Maurtuaa 2014, Habib 2014)),

(2) Ground robots inspecting structured facilities along established concrete access roads (e.g. onshore gas-and-oil facilities (Soldan 2012, Barber 2015, Steele 2014, Shukla 2016a)),

(3) Mobile (ground) robots inspecting facilities over a 3D maze of gangs (offshore oil platforms (Chen 2014ab, de Carvalho 2013, Shukla 2016b)),

(4) Wall climbing robots inspecting the 3D structures (Leon-Rodriguez 2013, Maurtuaa 2014)),

(5) UAV/AUV robots inspecting the consistency of vertical structures from the air or under the water (e.g. tanks, vertical piping (Cacace 2014)),

(6) “Pig” robots inspecting the insides of the piping (Shukla 2016a), and

(7) Cooperative teaming of a number of such robots (Surmann 2008, Kyrkjebo 2009).

The co-worker robot of SINTEF is designated to work on offshore gas-and-oil platforms, but will be developed and tested primarily in the onshore-like lab facilities (R5-COP D41.10, D41.20). In the present report we solve thus the planning problem for the case (2) “ground robots inspecting structured facilities along established concrete access roads”.

The key rationales of such application are the utilization of the potential of sensor fusion on a server platform placed close to the process, minimizing the need for multiple costly fixed sensors, the autonomy management in task solving without being dog-lead by a human technician, advanced visual capabilities, capability of knowledge intensive data processing,

[1] In the report this robot will be referred as CFR - Co-worker (and First Responder) Robot, considering that one of its roles is to act also as the first responder (for more detail see (R5_COP D11.10)).

re-configuration based adaptation in various environmental conditions, and even the prospect of working and teaming up with other robots, or in a mixed robotic-human environment. However this last issue hasn't been seriously researched yet due to the difficulty of the theoretical and legal problems (Habib 2014).

Even in case when only a single ground based CFR robot is involved, its application domain is a complicated problem domain posing multiple design challenges (Kyrkjebo 2009, Kroll 2008, Graf 2007, Distante 2009, Maurtuaa 2014, Sensabot 2012, Shukla 2016ab, Soldan 2012, Steele 2014, Chen 2014ab).

The CFR must be able to perform and to plan for a wide spectrum of activities, like moving around, measuring, monitoring, communicating, making maintenance, ranging from remote visual inspection to direct physical intervention. These duties may differ depending on the situational context, e.g. in a normal regime vs. in an emergency regime, and may depend also on the level of the autonomy available to the robot (from the teleoperation to a full autonomy.

The working environment of the inspecting robot is partly static, partly dynamic, with widely variable external weather conditions. The outdoor environment is structured. Onshore there is a lattice of wide access roads, smooth road surface, clearly defined areas. Offshore there is a lattice of wide narrow passages, plain steel floors and gratings – often with small holes, sharp edges, slopes and steps, maze of piping and equipment.

Robot actions are immersed in human traffic environment, with expected cooperative behaviour (robot-human) at mission/task/maneuvering levels. An interesting aspect of the robot activities is its variable adaptivity. In the normal regime the CFR has to adapt to the pedestrians, in the emergency regime it is rather the pedestrians who have to adapt to the CFR movements.

Mission critical character of the co-worker problem domain, performing timely duties in a mixed human-industrial facility environment imposes specific requirements for the system hardware and software (see (Chen 2014ab) for the general guidelines, and (R5-COP D41.20) for the more specific CFR related requirements).

With its hardware configuration (mobil base, sensors, actuators, communication facilities) the robot must be well adapted to the movement in the structured access road environment, must perceive its surroundings, especially to detect obstacles, must precisely track its position (high precision navigation for autonomy), and must be equipped with appropriate application sensors and tools required for the autonomous or teleoperated inspection and manipulation.

Hardware configuration must be coupled with the software providing functions to navigate the inspection sites without collisions in the teleoperating mode, and in the automatic mode (moving to a given target autonomously) and execute pre-programmed inspection and monitoring tasks automatically (with alerting the remote operator when abnormal sensor values are detected). This ability must be backed by the autonomously planning and following a (optimal if feasible) path to a given target, based on an environment map containing obstacles and free passages in the environment.

Considering in-field usage of the robot it should be maitenable without expert knowledge, easily and intuitively as a daily-used tool (new inspection and manipulation tasks can be programmed quickly and without the assistance of specialists, anyone working next to the robot can interact with it safely).

The more specific requirements and robot related data can be found in (R5-COP D11.10, D41.10, D41.20). From the point of view of the design of the Planning and Reasoning Architecture we only add here that the robot environment is not heavy in obstacles (but adverse weather conditions can also build up obstacles: water pools, snow mounds, etc.), dynamic obstacles are only solely humans, or human driven vehicles, who can (or should) partly adapt to the presence and the movement of the robot. The robot is embedded in the human traffic and everybody should observe the rules of the obligatory and the good road behaviour. There are no closer encounters with the humans, other than to avoid them. Pedestrians and bicyclists must be avoided, navigated around, etc. in a "friendly" way. The robot should in general behave to the human by-passers in a predictable way. The danger of collision should be mitigated. Finally, the robot is subjected to a non-continuous working regime, with maintenance and battery loading breaks between the inspection missions, performed in a controlled garage environment.

3 Co-worker robot planning problem

The co-worker domain means a complex planning problem and its complexity stems from the fact that the robot duties and the related activities:

- Vary in their time span and the durability of their effects (long term actions, short term actions),
- Vary in the geographic range of their impact (local actions and actions over a distance),
- Are distributed across the components of the plant information system (the CFR and/or the operator system),
- May simultaneously involve different robot hardware components (e.g. sensing actions are independent from movement actions).

Furthermore the full inspection plan of the industrial facility is necessarily an integrated hierarchy of plans because in general:

- The industrial installation to be inspected is composed from the hierarchy of separately placed process installations, connected with transit access roads, used jointly with the human traffic.
- Each process installation has multiple equipment points to be inspected, accessible locally, possibly away from the transit roads.
- States and readings of the particular process equipment points may be functionally related (i.e. usually all equipment of a given process should be inspected before moving on to inspect another process).
- Each equipment point requires specific (measurement, maintenance) operations.
- When inspecting a process a particular equipment point may be designated to be inspected first, as the representative of the process state, complicating thus the action management.

3.1 Planning problem decomposition and plan integration

The overall CFR planning problem can be decomposed approximately into the levels of:

- Planning the mission: the global movement over the whole facility,
- Planning the inspection: the sensor/actuator related activities in specific places along the mission path, and
- Planning the movement: the local movements negotiating the distances between the inspection places along the mission path.

In the following we analyse shortly the main design and programming issues pertinent to each step in the planning hierarchy. More detail will be given in Section 4.2 when the architectural elements responsible for the planning levels will be described.

3.1.1 Planning the mission

Mission planning involves the global plan of the whole inspection. It tells in what order to access the process installations, and in what order to inspect the process equipment at a given installation. Furthermore it specifies the segments of the access roads connecting the inspected equipment.

A list of equipment to be inspected (together with the possible constraints) is given by the operator (Fig 1)[2]. It is based on maintenance strategies, safety regulations, productivity guidelines, etc. Then the mission planning means:

- Fetching the localization of the equipment from the knowledge base,
- Fetching the lay-out (localization) of the access roads from the knowledge base,
- Computing a movement plan along the access roads, visiting the equipment places, and observing the possible constraints on the order of the visiting, etc. (Fig 2-3)
- Checking this plan for time/battery constraints, and replan if necessary (Checking the plan not only should take into account the cost of moving around, but also the cost of making inspection at the equipment places, e.g. the power requirements for measurements, the time to make measurements or maintenance. Such cost estimates can be precomputed and put into the knowledge base priori to planning the mission.)

As the particular process/equipment needs to be inspected only once during an inspection tour and that the tour should be kept within the battery life limits, planning the mission schedule has a character of the Traveling Salesman Problem (TSP) (Gutin 2006). The possible constraints on the ordering of the visiting equipment make this a problem similar to the Clustered (or Colored) Traveling Salesman Problem (CTSP). In the CTSP the clusters (or colors) allow the planner to group inspection points and specify desired constraints on how the robot should carry out the inspection (Edelkamp 2015, Li 2015). In particular, the robot is required to inspect all the points in one cluster (or color group) before inspecting a point region from another cluster (or color group).

An optimal mission plan may mean the fastest route, the shortest route, a route avoiding access lanes frequented by motor vehicles, by bicycles, by walking pedestrians, etc. Due to the structured and obligatory character of the access roads (movement possible only along the access roads) not any kind of path optimality can be taken into account. Further important complications to planning the mission could be:

- The possibility that the inspection order of the process sites, the inspection order of the equipment points at a given process site may be partially set by the operational require-

[2] This concise formulation may cover also more involved plans, like to inspect certain points, go to the garage to recharge batteries and continue, or that the robot needs to be re-equipped with different sensors to inspect different elements, and that might mean to move back to the storage area to change sensors.

ments, and these may vary with time. Operational requirements may also require to selectively inspecting only a designated part of the equipment points.

- The distance between the inspected equipment points may not be metric (Euclidean) but governed by the network of connecting access roads[3].

- most of the inspection points can be accessed in multiple ways, and certain access road segments must be reused in the interest of the reachability of all of the inspection points. (It means e.g. that the TSP kind of algorithm cannot be run directly on the true access road graph. Such graph must be transformed first into a „shortest access path”-kind of graph, running e.g. the A* search between any two inspection points in the original problem graph (Russell 2005). The optimal mission plan can be computed also directly based on the true access roads, using other more involved algorithm).

- The time/battery cost of the inspection measurements must be calculated into the computation of the shortest mission route (staying in place and making measurements also uses up time and energy).

A specific mission-like plan is the emergency access plan to direct the robot to a place of the alarm the shortest possible way. To this aim a simple A* search will suffice, run on the true access road graph, with modified edge costs (without time/battery cost of the inspection measurements along the path)[4]. A possible option could be to use faster ε-optimal A* (anytime) search with a non-admissible heuristic or hybrid solutions integrating the path movement with kinematic constraints (Thayer 2010). In case of the inspection site of a very low complexity (only some places to visit) such emergency plans could be even precomputed for strategic positions and destinations.

[3] There could also be situations where there are no designated roads, but the robot is able to plan trajectories freely through the environment.

[4] The basic A* algorithm works in the discrete state space and does not take into account the kinematics and/or noholonomic constraints of the robot. The global A* level planning must be coupled to local trajectory planning (see e.g. the ROS Navigation Stack, Section 4), or a more advanced integrated algorithm must be used, like e.g. the hybrid A* of (Petereit 2012).

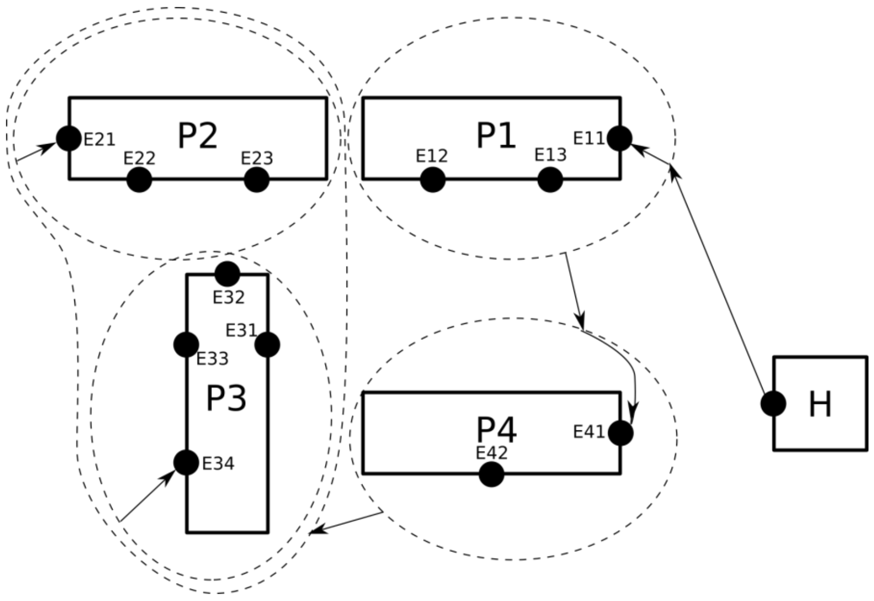


Fig 1. Functional Mission Plan with mission (inspection) goals given by the operator (hypothetical installation, graphic view). The broken line encapsulates functional clusters to be inspected completely before moving on to the next cluster (P – Process, E – Equipment, H – Hut). Arrows indicate priorities (ordering of the clusters, first equipment to be inspected in a cluster, etc.). The CFR is to inspect the P1 installation first, starting with the equipment E11. Ordering of the other equipment here is free. Then the next installation to visit is P4, with the first equipment to be inspected being E41. The inspection order of the remaining P2 and P3 installations is free, with the constraints that in P2 the first inspected equipment is E21, and in P3 E34.

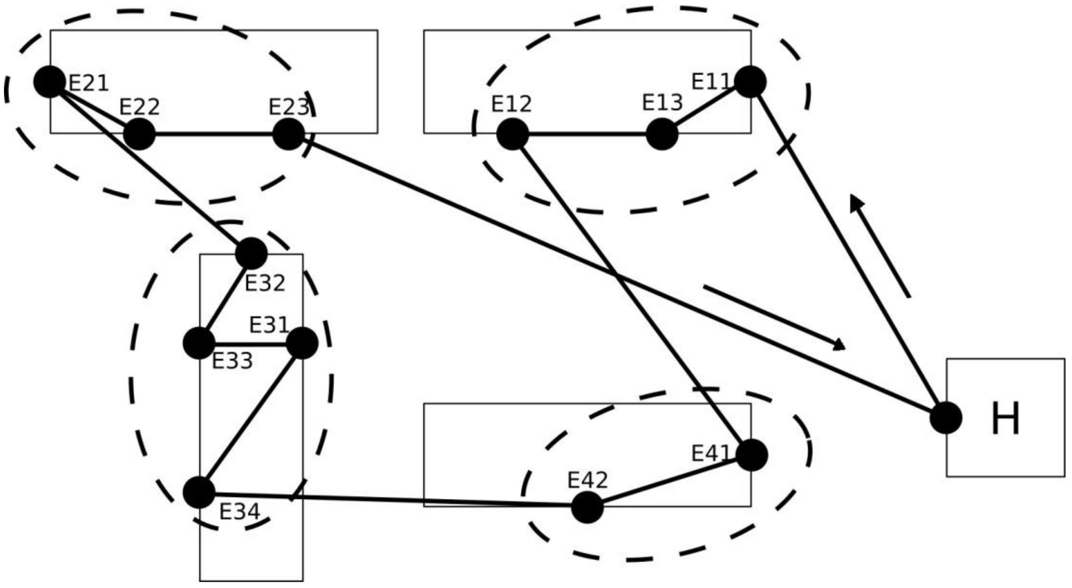


Fig 2. Optimal scheduling of the Functional Mission Plan from Fig 1. The broken line encapsulates functional clusters to be inspected completely before moving on to the next cluster (P – Process, E – Equipment, H – Hut).

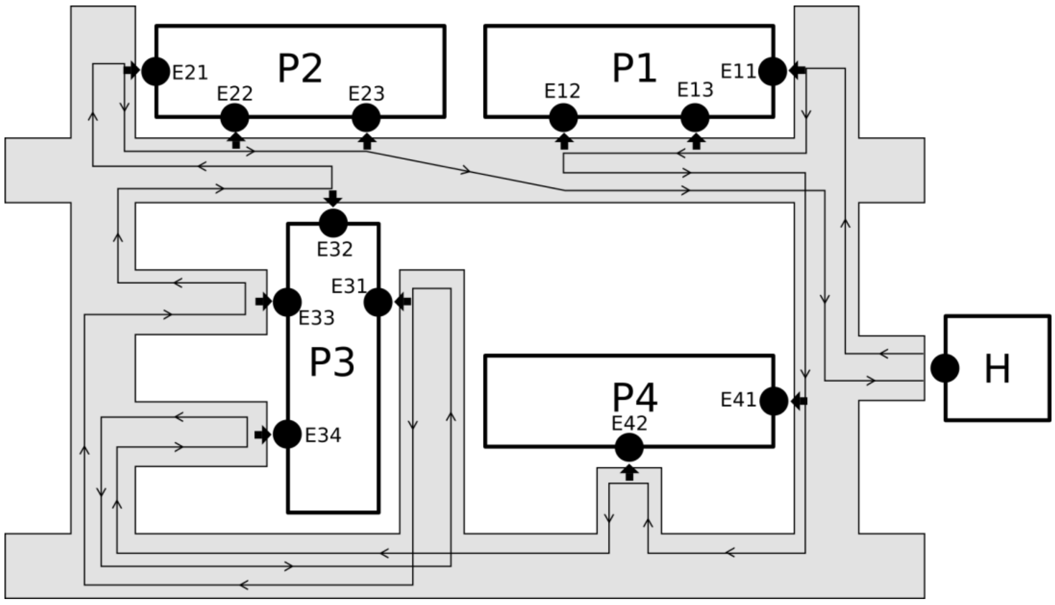


Fig 3. Metric Mission Plan – scheduled real movement of the CFR along the access roads. The thick black arrows designate Inspection Plan Stubs at the equipment inspection points.

3.1.2 Planning the inspection

In the robotic literature inspection planning is almost solely meant as the visual coverage planning to "see" the whole area of the given workspace by a mobile robot (Faigl 2011, Englot 2012, Galceran 2013, Arain 2015ab, Pierce 2014, Reyes Ballesteros 2012). To this aim a number of optimally placed viewpoints must be chosen, with the specified locations and poses for the robot to direct sensors to assure the full (or maximized) area coverage (so called Art Gallery Problem, https://en.wikipedia.org/wiki/Art_gallery_problem). The essence of such inspection planning is really the robot motion, as the visual inspection (maintaining cameras and running image processing) does not add essentially to the time and energy costs.

In case of the CFR robot the situation is different. The inspection activities do not only mean the usual visual (perimeter, area, intrusion) safety inspection performed by scout robots. For the inspection basically four kinds of inspection (and maintenance) duties can be assumed:

(1) Visual inspection on the fly with the video equipment used in parallel with the robot movement (e.g. along the wall, a pipe, a road). The sensory (image) information enters the stream of continuous image processing, object recognition, etc. to sensor fusion. There is no need for separate Inspection Plan Stubs.

(2) The robot has to stop, perhaps fine positioning itself (exact location, pose, direction), then has to direct sensors (pan-tilt), and run sensor control programs. Such programs must be derived from sensor related inspection plans.

(3) With sensors mounted on a fully functional arm, possibly with grippers to collect samples, beside fine positioning and sensor control actions, arm positioning action plan is also needed.

(4) Full maintenance with a hand-like gripper arm or specialized grippers (valve opening/closing, switching, etc.). Here a full plan segment is required to handle the arm manipulation.

Based on the proposed hardware architecture of the CFR presented in D41.20, we assume that options (3-4) are not foreseen in the final architecture, option (1) is automatically performed by the command and data flow in the robot general software architecture (Fig. 5), and it is option (2) which require further detailed analysis.

Inspection planning involves activities that happen at a single inspection point of the mission plan. In principle these are the measurement and/or monitoring activities, CFR-to-operator communication, and some local maneuvering to get into the position and the pose required by the sensing or by the actuating protocols. Inspection planning should provide also a time/battery estimate for the mission planning, but otherwise it is an independent planning activity and algorithms.

Due to these reasons the inspections duties cannot be optimized like the Art Gallery Problem view points. Changing weather, winds, specifics of the sensors make it difficult to specify (and to optimize) gas leakage detection places, especially when connected with the inspection places functionally tied to the location of the particular equipment hardware (for similar problem, but limited to the gas leakage sensing, see (Distante 2009, Arain 2015ab).

Inspection planning (in case of particular equipment) must observe the testing and the monitoring protocols prescribed for the inspected equipment and also the calibrating and the measuring protocol of the used sensors. This information belongs to the functional description of the plant equipment and the robot peripherals and is (should) be stored in the CFR system Knowledge and Data Base (KDB).

Considering that the selection and the ordering of the inspection activities at an inspection point are highly dependent on the function of the inspected equipment, the inspection planning is heavily domain and application dependent and the literature on such inspection planning is virtually non-existent. Perhaps the most relevant project of the similar aims – MAINBOT (Mobile robots for inspection and maintenance activities in extensive industrial plants, project reference: 285245, closed in 2014 (Maurtuaa 2014) tells almost nothing of how the inspection activities were programmed (planned) to be intertwined with the movement along the access paths. It seems that at present the focus of attention is to solve the sophisticated navigation problem and design sensors specifically mountenable on the mobile autonomous robots.[5]

Due to the natural structuring of the inspection duties into the hierarchy of the ordered tasks, perhaps the most suitable representation and the planner may be some version of the Hierarchical Task Networks (HTN) (representing and decomposing the compound activity as a conditioned network of the more primitive tasks, until the level of directly applicable operators is reached). Generic HTN plans can also be neatly stored in the equipment description slots of the CFR (frame based) Knowledge and Data Base (R5-COP D26.21). Plans for the same equipment at different process sites can be reused by parametrizing. Generic plans applied

[5] To mount a portable sensor on a pan-tilt column is only a part of the solution. Portable sensors are usually designed to involve an active participation of the human operator (pointing, checking, maintaining). Such functionalities must be fully automatized and made programmable.

to similar, yet differing equipment or situations can be tailored before being fed to the planner.

Inspection planning means filling-in the mission plan with the measurement and maintenance actions when the co-worker arrives at the end of the road segment leading to equipment. We assume that the time and energy spending for these activities was taken into account in the mission planning (based on heuristic estimates, earlier observed or learned data), but now the appropriate control actions must be listed to be passed later to the control hardware.

3.1.3 Planning the movements

Movement planning covers all the locomotion and obstacle avoidance control when getting from one inspection point to the next, negotiating the necessary road segments. Considering that the environment is highly structured (no free-country dirt roads, but usually straight-line concrete paths and crossings) the principal problem is to percept and to monitor unexpected structural obstacles and properly interact with the human traffic along the movement path.

The goal should be to ensure that the CFR movements are (Akin Sisbot 2007, Chen 2014ab, Shukla 2016ab):

- „Physically” safe (bring no harm to the human),

- Reliable and effective (propel the robot along the inspection route),

- „Mentally safe”, i.e. socially acceptable (taking into account the way the human move and also the human preferences and needs).

Various human aware motion planners, human aware navigation methods are reported in the literature, working e.g. with safety and visibility zones ensuring that the robot does not move too close to the humans, remains as visible as possible along the path, and is not too close to the humans when it appears in their field of view (Bogomolov 2003, Shi 2008, Buisson 2013, Cho 2010, Guzzi 2013, Hamasaki 2011, Moussaid 2011, Pacchierotti 2005, Tamura 2010, Vasquez 2013). Interesting is also intention-aware pedestrian avoidance strategy estimating and projecting first the movement intention of the encountered pedestrian and adjusting the robot movements accordingly (Akin Sisbot 2007, Bandyopadhyay 2013).

In the CFR setting it is important that the movement space is usually heavily constrained in one dimension (path width vs. path length) and that the usual lay-out of the access roads is regular. It means that basically there are two situations to handle: pedestrian or cyclist crossing (i.e. passing across the movement path of the CFR in the road crossing), and approaching (coming from ahead). Furthermore the CFR in a crossing can pass straight across, turn right, or turn left, and along a straight lane it can go ahead, overtake, or make a U-turn. The regularity of the paths can make it easier to recognize reliably especially the cyclists, who move faster, maneuver less, and suffer more from a possible collision.

An important aspect of the CFR human obstacle avoidance is also the fact that the robot moves around in a restricted area and the encountered pedestrians or cyclists are fit professionals who are expected to be instructed on the presence, the behaviour and the capabilities of the robot, making the human more predictable.

An interesting option fusing the human obstacle avoidance with learning is to collect a data-base of the human's movement tendency (velocity vectors) at the places of particularly high collision risk and to use it to predict the behaviour of the encountered passers-by (Hamasaki 2011).

The output of the movement planning are thus the velocity commands to the motor control related to the obstacle avoiding motion trajectory consistent with the kinematic properties of the robot.

An important issue of the movement planning to be solved by the planning architecture is the fine maneuvering problem of getting into the position (and pose) for the inspection measure-ments. The position/pose to activate the inspection sensors should be specified in the equipment descriptors in the Knowledge and Data Base. However even if this position/ pose is given as a goal for the movement trajectory, the true final position (and pose) of the ap-proaching movement can differ too much for the measurements to be admissible. The CFR should be able to fine maneuvering into the position, using movement commands perhaps different from those permissible in the large scale trajectory computation (the primary issue is the permissible sideward and backward motions).

Finally plenty of papers deal with the area inspection planning, decomposing the problem hierarchically from optimal global paths to the local motion, suggesting many useful or inter-esting solutions. Worth reading are (Geraerts 2007, 2010, Likhachev 2010, Englot 2011, Pe-tereit 2012, Ulrich 2016).

4 Co-worker Planning and Reasoning Architecture

As a remote sensor and maintenance platform the co-worker information system must be integrated into the whole plant information system (see Fig 4). Co-worker accepts tasks, guideline information, commands from the higher levels of the system hierarchy (Operator Center overseeing and scheduling the CFR tasks in the mission area, Remote Operator Center overseeing the production in plant units within the whole industrial installation) and provides them with the actual process data.

Although a seamless planning and execution of the mission is primarily the responsibility of the co-worker system, the support given by the higher levels of the system hierarchy is essential. We will characterize shortly the assumed responsibilities and interactions between the Co-worker robot, Operator Center Operator, and Remote Operator systems focusing on the planning related issues.

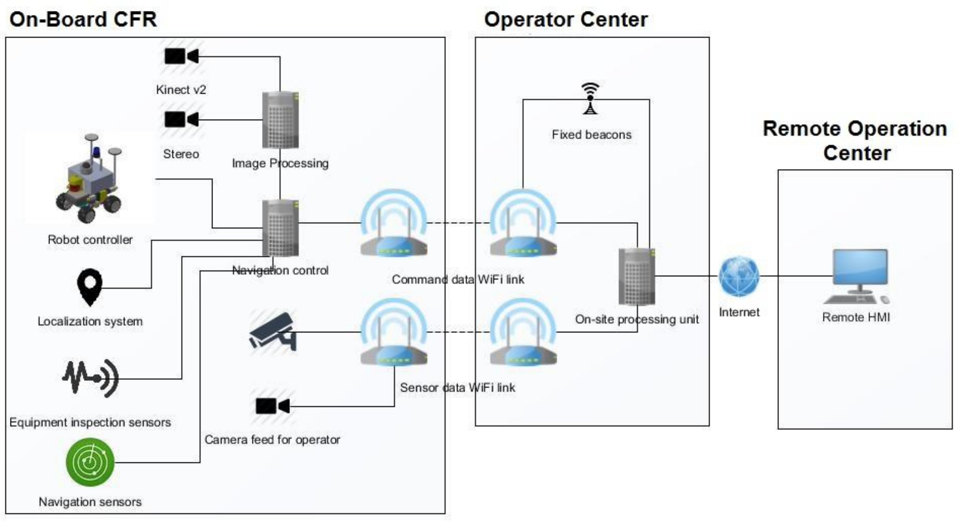


Fig 4. The rough view of the distributed information system including the co-worker First Responder. Note the different communication channels for the video stream and for other data (from R5-COP D41.10)

Remote Operator Center (ROpC)

Remote Operator Center provides general purpose information, like regulations and guidelines how to organize and report the inspection missions, weather forecasts, repositories of manuals, data, and algorithms, etc. It is the destination place of the report summaries and critical event logs.

Operator Center (OpC)

Operator Center helps to organize inspection missions in the supervised area by maintaining repositories of local information (e.g. various maps, equipment models, protocols of the inspection procedures, etc.). Staff of the Operator Center is responsible for the selection of the inspected equipment and the ordering of inspection points (or at least approving the selection done at the on-board CFR system). The Operator Center collects and post-processes the inspection data, and extracts information essential for the proper running of the industrial installation. In cases when the circumstances surpass the autonomous capabilities of the co-worker, operator resident in the Operator Center may resort to teleoperating, overriding the planning with direct movement and measurement commands and evaluating the situation personally.

On-board CFR robot system (CFR)

Prior to the inspection the on-board CFR system is involved in planning the inspection routes (and missions) for particular equipment listing required or approved by the Operator Center.

During and/or after the normal inspection CRF is monitoring on-site conditions, collecting/ summarizing/ reporting (essential) inspection and other data up the operator hierarchy, and re-planning the missions in case of reported durable/permanent traffic obstacles that the CFR maneuvering capabilities cannot overcome.

During the emergencies CFR is monitoring conditions on-site, reporting (up) conditions on-site, accepting commands directed it to the emergency area, and enabling telepresence for the operators from the Operator Center by teleoperation.

4.1 CFR Software Architecture – General Remarks

We focus now on the on-board software system of the co-worker. The proposed functional architecture can be seen in Fig 5. It follows from the general Sense-Plan-Act and the Layered-Control paradigms (Mataric 2002, Kortenkamp 2013). To control the robot information from and about the environment (robot included) must be collected via sensors. Processing this information with respect to the robot goals leads to decisions about how to act. Finally decided actions must be executed in the robot environment.[6] The vertical hierarchy separates also the robot processes acc. to the time-scale. Low levels, with fast algorithms working on numerical data provide reactive real-time performance; higher levels working more slowly on mixed numerical-symbolical data provide flexibility and robustness. Other more specialized modules account for the specific inspection duties and the teleoperating option.

The proposed CFR architecture is presented in Fig 5. Comparing to the Fig 4 of the all-plant information system, Fig 5 shows only the CFR (on-board) modules and those modules of the Operator Center, which are relevant to he functioning of the Planning and Reasoning Architecture of the CFR. In the present report we had to assume that the information systems of the Remote Operator Center and the Operator Center are already designed and operative (after all an autonomous CFR is an add-on to the information system of the industrial site), and at most the Operator Center admits some modification (to accept the CFR data and to control the CFR movements, if needed).

[6] It is worth to mention that this sense-process-act loop is nothing else as the general “action-loop” of an agent system (Russell 2005).

In the following (in the alphabetical order) we will shortly characterize the function of every module and explain the interactions between them (see also (R5-COP D41.20)).

Battery Status Module provides periodical time stamped status of the battery voltage level for the Sensor Fusion Module and further for the Mission Control Module. This data is logged periodically in the CFR Knowledge and Data Base (KDB) and is essential for early decisions influencing the course of the inspection towards avoiding stranding the CFR on the site.

Communication Modules. One Communication Module is responsible for transmitting the data from the OpC to the CFR (prior to the inspection), occasional data and events from and commands to the CFR (during the inspection), and a fuller data record (if required) after the inspection is concluded. The particular identity of the relayed information and the communication schedule depends upon the situation recognized by the Mission Control Module. As a rule all abnormal information is stored and real-time transmitted to the operator (if the observation coverage allows). The second Communication Module serves as a teleoperator video stream channel.

Emergency Signaling Module controls and processes audio and visual warnings activated with the commands received from Mission Control Module (autonomous functioning after alarm is sounded) or from the Teleoperator Module (if the operator decides to sound alarm when teleoperating).

Emergency Status and Control Module encapsulates the hardware used for emergency signaling (to and from the environment, like e.g. warning lights, and robot emergency stop button). Warning Sound and Speech Generator plays audio warnings in case of emergency. Warning Lights Control controls the warning (blinking) lights used in the overtaking maneuvers and the emergency.

Image Processing Module embraces algorithms needed to transform and process visual data (scanner data, point clouds, object independent features, object related features, etc.), depending on the image sensors used and the tasks demanding visual information. It acquires the rough data from the Visual Data Acquisition Module and provides processed information for the navigation and obstacle avoidance and to support the preparation of the decision making. Image Processing Module connects also to the KDB for fetching (optional) image patterns (used in object recognition) and for (optional) storing the images to reference particular events.

Inspection Measurement Status and Control Module groups software modules wrapping and handling installation related sensors (gas leakage sensor, flame detector, etc.). The sensory data is directed to the Sensor Fusion Module, the sensor control commands come from the Inspection Planner (autonomous inspection) or Teleoperator Module (during the teleoperation).

Inspection Planner is responsible for the integration of the routing plans with the inspection actions demanded by the visited equipment. Upon the execution of the mission plan the Inspection Planner directs (road segment) movement commands to the Navigation Planner and the measurement and maintenance commands to the Inspection Measurement Control (Fine maneuvering commands required to get into position for the measurement are also directed to the Navigation Planner).

Knowledge and Data Base (KDB) is a storage system which stores static information, like installation functional structure, superstructure and road lay-outs, equipment descriptions, mission plans, various maps, global parameters, algorithm libraries, and dynamic run-time status information, measurements and events, furthermore images, possibly also audio and video records. From the content of the KDB the Critical actual information is directed to the operator, other data may be replayed to the Operator Center after the inspection is concluded, or may be deleted.

Learner Module is responsible for finding, identifying, extracting, and representing new and unusual information from the collected data, which may be of interest in the subsequent missions. The CFR will be able to learn from past actions and continue the mission without requiring frequent operator's intervention.

Localization Module collects and integrates data from localization (GPS) and odometry sensors (but may also utilize landmark image information).

Location Status Module brings in the time stamped GPS data for the localization purposes (for the Localization Module).

Mission Control Module is the main decision-maker, which accepts mission goals, manages global parameters, interprets events and data, decides on storage and communication with the operator, computes the mission plans, based on the goals, and sets accordingly the functional options in other modules. Mission Control Module connects to an optional Operator User Interface operational when the CFR is maintained in the garage.

Motion Status and Control Module encapsulates motor drives and low level reactive motion control. It provides robot (motor) diagnostic and status data for the Sensor Fusion. Accepts velocity controls from the Navigation Planner.

Navigation Planner is responsible for the non-colliding execution of the high level movement commands, respecting the robot kinematic. It contains global and local planners for the motion phase of the CFR activities, acc. to the occupancy and cost maps acquired from the Storage, actual percepts, and motion direction from the Inspection Planner.

Odometry Status Module provides periodical time stamped pose and speed data for the localization purposes (for the Localization Module), and for the general status of the system (for Sensor Fusion).

Sensor Fusion Module processes sensory data and actuator status data into log information and events (based e.g. on the limiting values provided from the storage). It accepts various kind of data, evaluates them for normal/abnormal status, identifies events, qualifies events toward triggering information, and prepares a package of actual information for the Mission Control Module to make decisions.

Teleoperator Module is responsible for by-passing the mission control and planners when the OpC operator is teleoperating. The teleoperator exercises direct control over the movements, measurements, image taking, issuing warning calls and switching on the lights.

Video Data Acquisition Module collects components related to the 2D or 3D information acquisition control (normal and IR cameras, pan-tilt units, scanners, mono, stereo, etc.) This data is optionally processed in the Image Processing Module (particular features or views are documented in the KDB, fused, or fed to the navigation control for low-level obstacle avoidance). Video data acquisition can be activated from the Inspection Planner or from the Teleoperator Module. In this last case, the video stream is directed straight to the Communication Module(2) which controls communication channel reserved for the teleoperating.

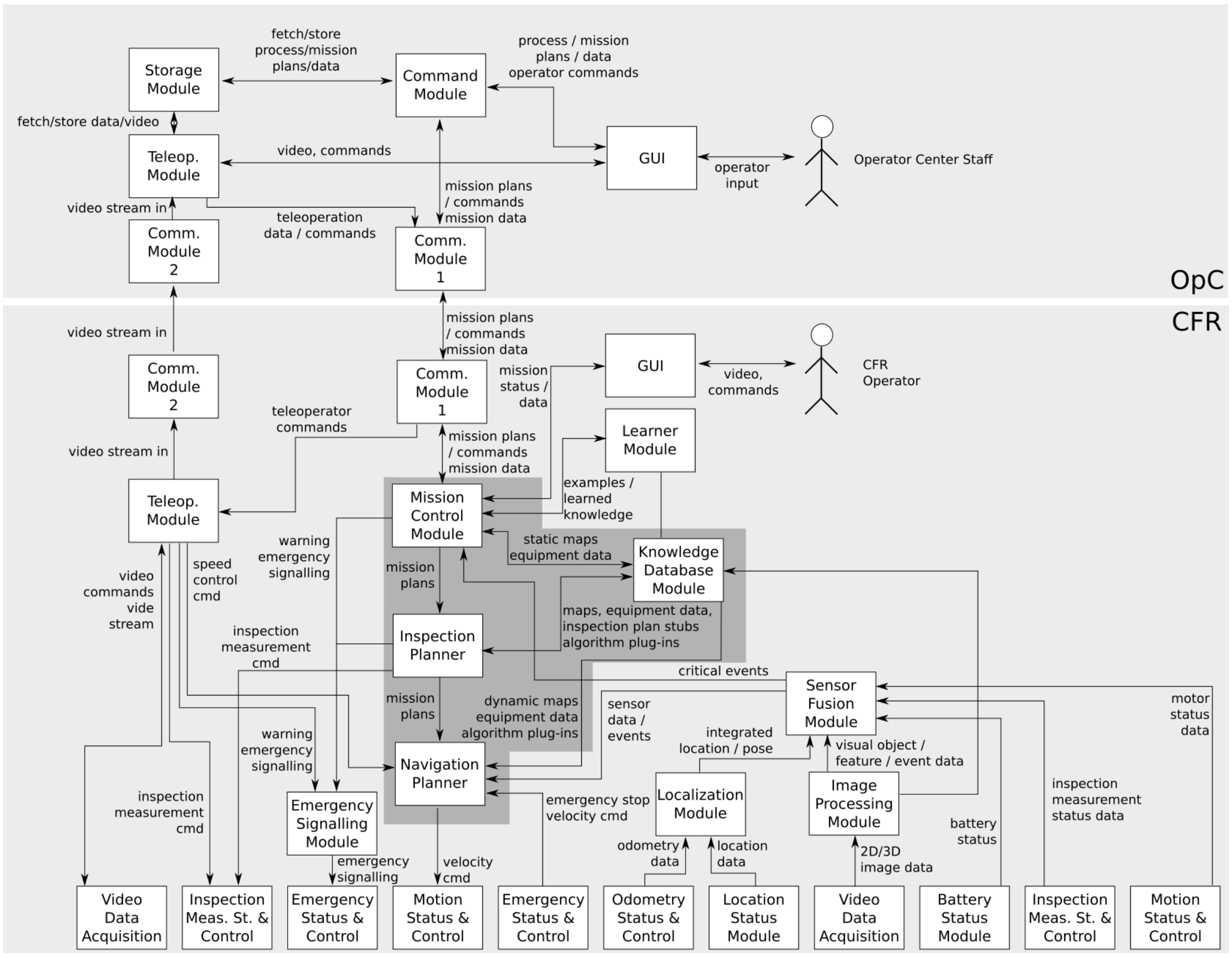
Inspection activities of the co-worker are supported by the Operator Center. For the development and testing purposes the supportive OpC functions can be realized on-board of the CFR co-worker, however in the final configuration they will probably be out placed to the Operator Center. To support the functioning of the CFR co-worker we would need:

Command Module (OpC-side) is the highest, possibly interactive (mission level) planner, setting the inspection patterns of the CFR around the plant, the order of the facilities to inspect, and the dependencies between the particular inspected facilities, all for normal and critical situations. Such pre-computed patterns are then communicated with the CFR Mission Control Module to initiate mission planning. The module should have an advanced GUI to facilitate the operator's planning work.

Communication Modules (OpC-side) are companions of the CFR Communication Modules, essentially with the same capabilities.

Storage Management Module (OpC-side) is the archive of the operator to store information relevant from mission to mission, to store maps, lay-outs, technical information, plans, confidence limits on measurements, etc. and finally also to store log data replayed after the inspection from the CFR to evaluate them, filter, learn from them, and extract for archiving and reporting.

Teleoperator Module (POpC-side) is the operator's end of the teleoperation command-and-data chain. Should be equipped with an advanced GUI for easy introduction of commands and an effective presentation (video, audio, feature maps, localization, etc.) of the CFR environment.



Storage Module
fetch/store process/mission plans/data
Command Module
process / mission plans / data operator commands
fetch/store data/video
Teleop. Module
video, commands
GUI
operator input
Operator Center Staff
video stream in
teleoperation data / commands
mission plans / commands mission data
Comm. Module 2
Comm. Module 1
OpC
CFR
video stream in
mission plans / commands mission data
GUI
video, commands
CFR Operator
Comm. Module 2
Comm. Module 1
mission status / data
teleoperator commands
Learner Module
video stream in
mission plans / commands mission data
Teleop. Module
examples / learned knowledge
Mission Control Module
warning emergency signalling
static maps equipment data
Knowledge Database Module
video commands vide stream
speed control cmd
mission plans
inspection measurement cmd
Inspection Planner
maps, equipment data, inspection plan stubs algorithm plug-ins
critical events
Sensor Fusion Module
motor status data
mission plans
dynamic maps equipment data algorithm plug-ins
sensor data / events
integrated location / pose
warning emergency signalling
Navigation Planner
visual object / feature / event data
inspection measurement cmd
Emergency Signalling Module
emergency stop velocity cmd
Localization Module
Image Processing Module
inspection measurement status data
battery status
emergency signalling
velocity cmd
odometry data
location data
2D/3D image data
Video Data Acquisition
Inspection Meas. St. & Control
Emergency Status & Control
Motion Status & Control
Emergency Status & Control
Odometry Status & Control
Location Status Module
Video Data Acquisition
Battery Status Module
Inspection Meas. St. & Control
Motion Status & Control

Fig 5. The software architecture of the Co-worker CFR. Shaded area is the Planning and Reasoning Architecture (the reason for the seemingly duplicated modules in the last row is a clear separation of the control (downward) and the observation (upward) information flow).

4.2 Planning and Reasoning Architecture

Planning and Reasoning Architecture of the CFR is composed from the Inspection Planner and the Navigation Planner Modules, supported by the Mission Control Module and the Knowledge and Data Base (KDB), see Fig 6.

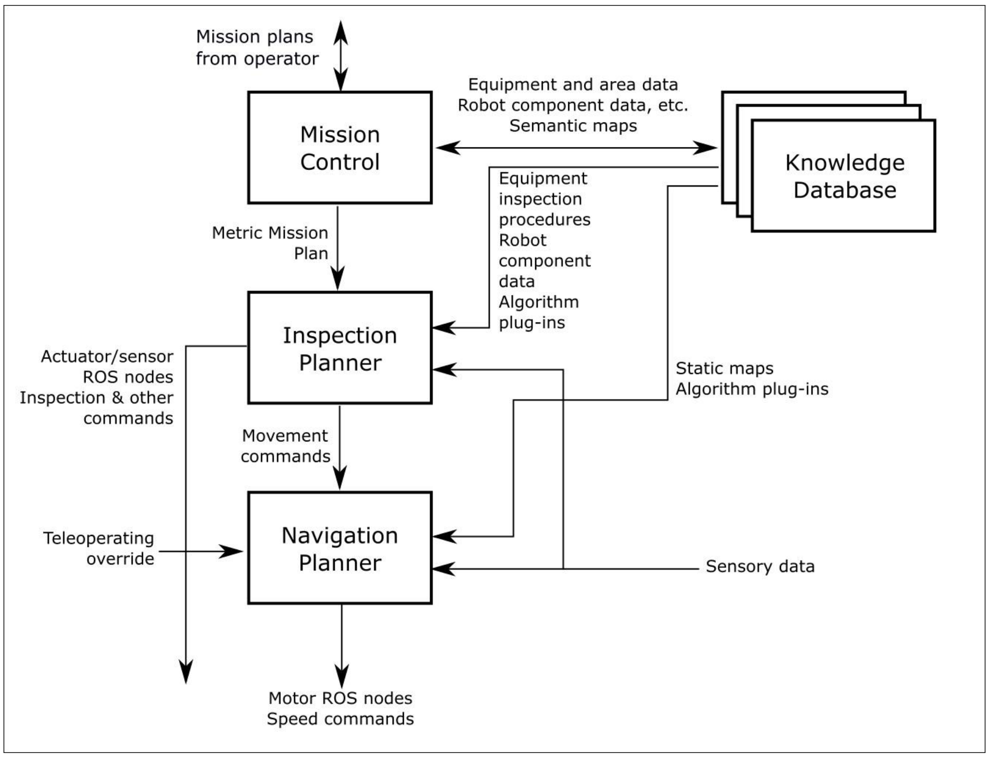


Fig 6. The Planning and Reasoning Architecture.

4.2.1 From the Functional Mission Plan to the Metric Mission Plan

Before the inspection starts Mission Control Module accepts (via the Communication Module from the Operator Center, or via the CFR GUI from the local operator) a listing of the equipment to inspect, with constraints (depicted graphically in Fig 1, referred as Functional Mission Plan (FMP)). Functional Mission Plan is stored in the KDB for reference and is recomputed into a Metric Mission Plan which takes into account the true distances and the access options in the environment. To this aim the KDB provides the information about:

- The structure of the installations and the dependencies between the equipment,
- The geographic localization of the equipment (semantic maps, KML format, see (KML, R5-COP D41.20) descriptors and the related bitmaps),
- The static access road map (and static permanent obstacles), with distances and localizations (in KML format, see (R5-COP D41.20)),
- The estimated costs of inspection at a particular equipment inspection point (estimated time and power consumption, which can be computed from the inspection protocols also stored in the KDB).

The Metric Mission Plan (MMP) is computed with the optional operator supervision acc. to the control flow in Fig 7. Information from the KDB is used by the scheduler algorithm to come up with an optimized schedule option. If the overall costs of the schedule (time, battery) are acceptable, the Metric Mission Plan is stored in the KDB and the control is passed to the Inspection Planner. If the schedule is too costly, the problem must be negotiated with the operator. The CFR requests the operator to change the inspection options (i.e. to come up with a new, less costly Functional Mission Plan) or to give the CFR a free hand in reducing the scope of the inspection (leaving out some of the equipment considered optional) until the costs of the computed optimal schedule will be acceptable.

Typical cases of the operator decision could be:

- No equipment is permitted to be left out (get a larger battery perhaps!),
- No equipment is permitted to be left out, but the scope of the measurements can be reduced (less time and energy),
- Operator lists equipment not to be left out, but the rest can be reduced (left out) at will,
- Operator asks for a proposition of a reduced plan and makes a decision (acceptance/rejection) after analysis, etc.

Please note that changing the scope of the visited equipment not only means that some path segments need not to be negotiated, but by influencing inspection constraints the whole schedule may change globally.

Computing functionally and geographically constrained optimal plan can be done with many algorithms. In concrete implementation the demonstrator can use here, e.g.:

- Optimized scheduler algorithms (see (R5-COP D26.30)),
- Globally optimizing clustered or colored Traveling Salesman Problem genetic algorithms (Hussain Ahmed 2014),
- Locally optimizing heuristic Traveling Salesman Problem algorithms (LKH) with false cost based clustering (Helsgaun 2006, 2014),
- Traditional search algorithms like A* (Russell 2005, ROS navfn, global_planner)[7],
- Sub-optimal (faster) versions of the optimal search, like anytime, weighted A*, etc. (Russell 2005, Hansen 2007， Thayer 2010).

It should be noted that due to the highly structured lay-out of the access roads in the inspection environment, this lay-out map in itself cannot be used by explicit search algorithms to compute the cost optimal schedules (it is strictly speaking not a graph from which a search tree could be computed). The access road map must be transformed into an abstract cost based access graph first. It can be done explicitly, running normal optimal search algorithm next, or it can be done implicitly being embedded into the body of the optimal scheduling algorithm.

[7] The basic solution built-in into the ROS global planner is the Dijkstra algorithm, producing the cost optimal path. The better alternative is to use A* search, providing the cost optimal solution, but with less (faster) computation. The difference is that Dijkstra algorithm uses only the cost estimate leading the computation occasionally away from the goal direction (if it seems temporarily less costly). Contrary to this, A* search uses also directional heuristics enforcing the search more toward the goal direction. If the used heuristic function is admissible (i.e. underestimating, like e.g. the Euclidean distance), A* yields the same optimal path as the Dijkstra algorithm.) A* is however provably optimally efficient, meaning that it computes the least number of nodes among similar optimal search algorithms (Russell 2005).

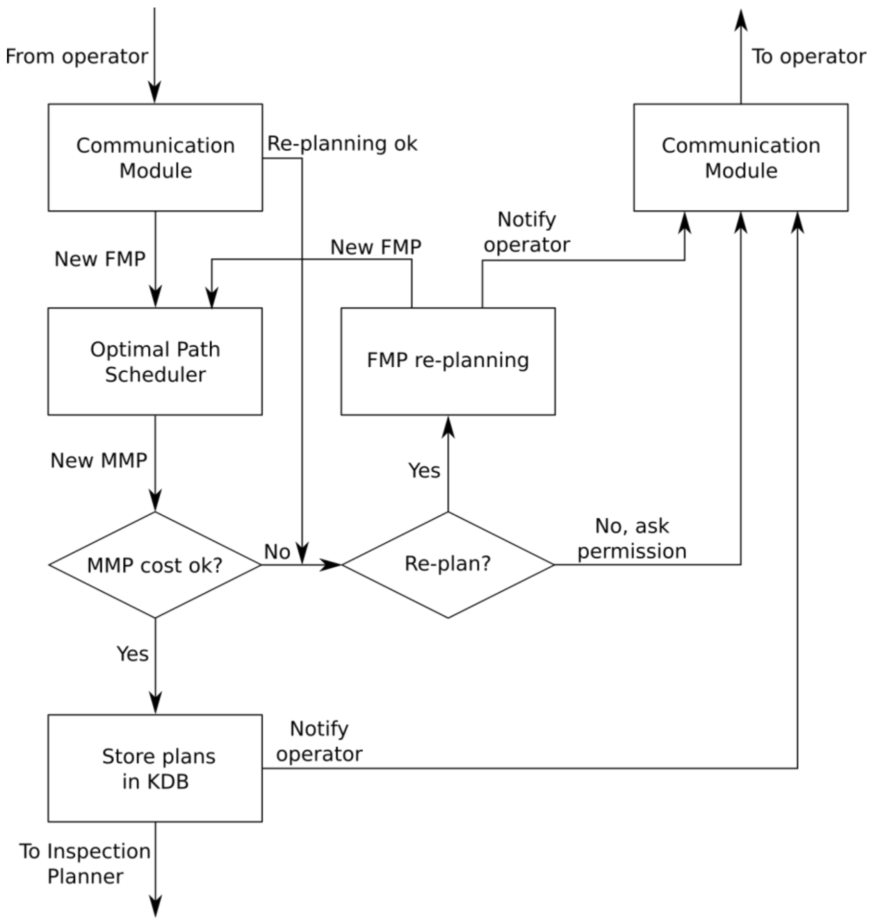


Fig 7. The control flow of the Mission Control Module.

4.2.2 From the Metric Mission Plan to the Inspection Plan

Assume that Metric Mission Plan of acceptable cost is stored already in the KDB and the control is now passed to the Inspection Planner. The Metric Mission Plan accounts for the approximate cost of the inspection actions, but as yet does not list them explicitly. It is now the task of the Inspection Planner to „fill-in" the Mission Plan with the real inspection activities and to control the execution of the whole plan, directing movement demands to the Navigation Planner and the other commands to the respective architecture modules encapsulating the inspection hardware.

In principle inspection planning must be done for every inspectable equipment (inspection point), by computing the plan stubs (plan segments responsible for the inspection, i.e. measurement, monitoring, maintenance activities) corresponding to the thick arrows in the Fig 3. Such plan fragments (Inspection Plan Stubs), conform with the regulations and the guidelines of the equipment management and of the robot sensors, may be then pre-processed, stored in the robot KDB, then at the suitable moment and location can be recalled and adjusted, or could be computed with the similar procedure in the real-time on the spot.

As mentioned before, the topic of the non-visual inspection plans may comprise diverse measurement modalities (temperature, water level, pressure, humidity, wind speed and wind direction, gas concentration and the shape of plume, burning flame, smoke, gas/fluid chemical identity, fluid spillage, etc.) and simply does not appear yet in the research literature. Papers reporting on the inspection robot prototypes dwell in details on the involved navigation and robot movements. Commanding the sensors is probably programmed manually or teleoperated, as nothing is mentioned of how such measurement plans could be formally derived (Arain 2015a, Barber 2015, Bennetts 2015, Codd-Downey 2014, de Carvalho 2013, Ibarguren 2013, Kim 2011, Kroll 2008, Maurtuaa 2014, Robogasinspector 2013, Soldan 2012, Steele 2014, TALON 2004). In the actual proposal of the Planning and Reasoning Architecture, regarding this aspect of the CFR planning we are thus entirely on our own.

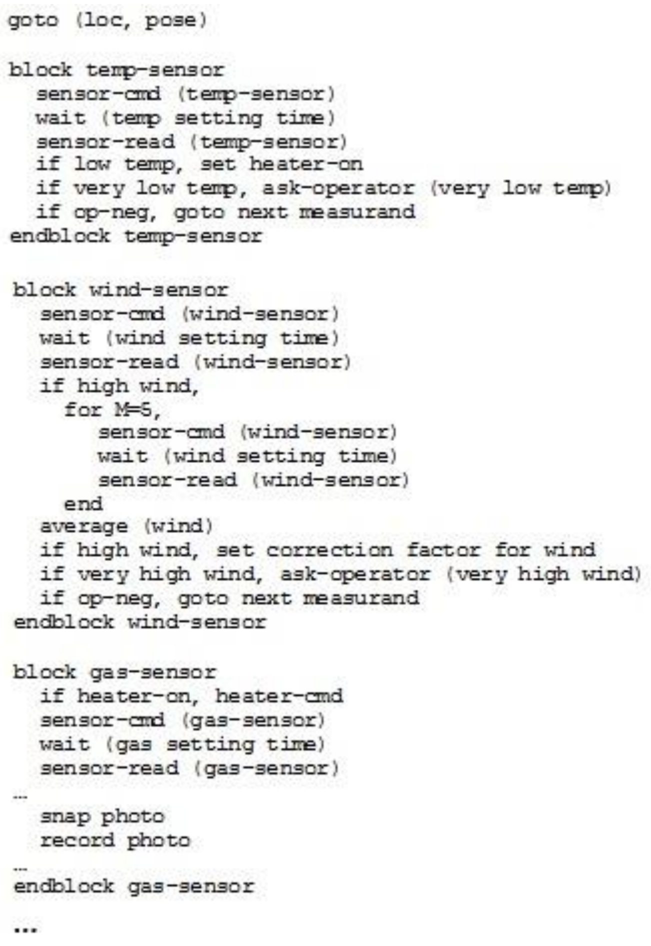
Inspection Plan Stubs

Inspection (plan stub) at a particular inspection point may cover diverse activities, like measurements, status and value checks, ask-and-acknowledge communication with the operator, local maneuvering, repeated and conditional activities, derived computations, result storage and reporting, etc. All these activities are functionally and hierarchically related (they apply to a particular hardware), are possibly partially ordered, and dependent on some free parameters (Liu 2012).

Due to this natural structuring of the inspection duties, the most suitable representation for them and the planner seems some version of the Hierarchical Task Networks (HTN) and HTN planner (e.g. SHOP, SHOP2, JSHOP2 (Nau 2003, Tian 2012)). Considering that the identity of the inspected equipment does not vary from a mission to a mission, this planning problem should be delegated to the Operator Center system and is not discussed in detail here. The preparation of the Inspection HTN plans for the diverse equipment belonging to the industrial installation is a task for the knowledge (metrology) engineers on the staff of the Operator Center (or even the Remote Operator Center). These prospects are discussed in more detail in the Section 5.2.

As mentioned earlier, the basic research in structured (HTN) inspection/ measurement task representation, sound choice of the primitive actions, and the modification of the HTN planners to cover such extension and to automatize the derivation of the inspection plans for a moment does not yet exist. At least no authoritative and useful theoretical publications appear. There is a very limited pool of available ROS packages with any kind of non-navigation planning (see (Cashmore 2015), also Section 5.2 and the discussion therein). The available publications on inspection robot prototypes keep back on it, and if the actual inspection runs are reported using named measurement equipment, we are forced to accept that the inspection planning has been done manually (Soldan 2012, Spiewak 2015, Barber 2015, Bennetts 2015, Habib 2014).

The structured HTN-like inspection plans (see Section 5.2) do not influence the functioning of the Inspection Planner. Inspection Planner accepts for the execution plans composed from primitive inspection actions, like e.g. the fragment of the inspection program in Fig 8. We assume in the following thus that the inspection plans were designed off-line at the Operator Center, were then processed from the highly structured HTN format to the largely sequential command format suitable for the Inspection Planner, or alternatively were designed manually, and were downloaded, together with other equipment descriptors to the CFR robot Knowledge and Data Base (KDB) before the mission, see Fig 10 (slots of Equipment). There such Inspection Plan Stubs will be fetched when needed, optionally parametrized, or simplified, and used to generate low-level commands (ROS Messages) to the inspection hardware.



```
goto (loc, pose)

block temp-sensor
  sensor-cmd (temp-sensor)
  wait (temp setting time)
  sensor-read (temp-sensor)
  if low temp, set heater-on
  if very low temp, ask-operator (very low temp)
  if op-neg, goto next measurand
endblock temp-sensor

block wind-sensor
  sensor-cmd (wind-sensor)
  wait (wind setting time)
  sensor-read (wind-sensor)
  if high wind,
    for M=5,
       sensor-cmd (wind-sensor)
       wait (wind setting time)
       sensor-read (wind-sensor)
    end
  average (wind)
  if high wind, set correction factor for wind
  if very high wind, ask-operator (very high wind)
  if op-neg, goto next measurand
endblock wind-sensor

block gas-sensor
  if heater-on, heater-cmd
  sensor-cmd (gas-sensor)
  wait (gas setting time)
  sensor-read (gas-sensor)
…
  snap photo
  record photo
…
endblock gas-sensor

...
```

Fig 8. A fictitious inspection program, which could be generated automatically from the HTN high level Inspection Plan Stub from Fig. 14. The plan orders the robot to maneuver close to the equipment, measure the ambient temperature and acc. to the readings prepare to heat-up the gas sensor, or to contact the operator if the cold is serious. If the conditions are mild or the operator gives it a go, the wind conditions are checked next. If the wind reading is high, a more accurate estimate is required, and the ambient conditions are similarly consulted with the operator. Finally, if the conditions are affordable, the very gas leakage reading is activated. Note the role of the control structures and the fixed and run-time control variables.

Executing Inspection programs

An important design step in developing good HTN representation of the inspection plans, but also when the planning is done manually, is the choice of the primitive action (HTN leaves, green level in Fig 14.). Primitive actions should be easy to be interpreted and executed by the Inspection Planner.

Inspection Planner fetches travelling commands from the Metric Mission Plan, passes them to the Navigation Planner in segments to move along, and when an equipment is to be accessed, the Inspection Planner suspends the Navigation Planner (see Section 4.2.4), fetches the right Inspection Plan (program), interprets its steps, and issues commands to other CFR modules.

The communication between the Planner and the CFR system will be based on 4 different types of messages (see in detail (R5-COP D41.20)):

Commands - are the actions that are dispatched to the robot system, and that the robot shall execute, e.g. “navigate_to(waypoint X)” or “capture_data(IR_cam)”.

Commands feedbacks - are linked to the previously defined commands. Most commands will require some time from the robot to actually execute the requested action. During the execution of the action, the robot system might send back information to the planner about the progress on the execution. This allows the planner to know when the action is performed, which is the result of the execution, or other information about the action that might enable the planner to take decisions.

Synchronous sensor/status data inputs are planner question about some sensor or status information. This will be done in a request/response way. An example of this type of communication is the planner requesting the batteries level issuing a "get_battery_info" request that will be replied with the corresponding battery level.

Asynchronous sensor/status data input is a unidirectional type of communication in which the robot system continuously publishes the updated information, and the planner can subscribe in order to get the updates. An example of this type of communication is the robot localization, which is continuously estimated and published, being possible for the planner and other clients to subscribe to it.

The design decision is the granularity of the primitive inspection actions (i.e. the relation of the green level and yellow level actions in Fig 14), or putting it bluntly: should the commands in Fig 8 be the ROS level topical messages, or may they be formulated more abstractly?

At stake is (1) the easiness of making the prototype (manual planning of the actions at the ROS level) and (2) the flexibility of the design. In this case the primitive actions should be defined simple, functionally close to the sensor and actuator level, yet flexible in the choice of particular hardware or technology (promoting using the same inspection programs with different implementation of the ROS nodes).

Actually the pursued solution is (1), using the following mapping to the ROS functionalities (see (R5_COP D41.20)):

Commands are realized as actionlib stack that provides a standardized interface for interfacing with preemptable tasks (ROS actionlib).

Commands feedbacks are provided by the actionlib stack with the Feedback and Result messages, which allow the server to inform the client about the progress of the action being executed.

Synchronous sensor/status data inputs are realized as ROS services - a pair of messages: one for the request and one for the reply. Client libraries usually present this interaction to the programmer as if it were a remote procedure call (ROS Services)

Asynchronous sensor/status data is realized as the stream of ROS topics intended for unidirectional, streaming communication (ROS Topics).

What would be needed to have option (2)? A kind of translating database providing for every primitive action a (short) procedure embedded in the body of the Inspection Planner module, and addressing particular ROS services, topics and messages, depending on the actual software and hardware configuration of the CFR.

Re-configuring the CFR will demand refreshing the translating procedures, but then the process of the derivation of the inspection programs from the inspection HTN representation will remain intact (see Section 5.2).

The possibility of the primitive action-to-ROS translation is optionally taken into account in the control flow of the Inspection Planner in Fig 9.

Inspection ROS packages

Regarding the poorly documented inspection plans in the reported robot prototypes it is instructive to review shortly the stock of the ROS packages relevant to the implementation of serious industrial structure inspection robots. The majority of the packages are toy applications, no packages fit to professional sensors demanded by the oil-and-gas application domain. Among ca. 90 packages (not everyone available) listed in (ROS Sensors), only ca. 9 packages are for non-motion, non-image related measurement sensors (ROS rosserial). The majority of them cannot sustain low temperatures demanded by the poor weather CFR requirements.

Ardusim (Arduino Sensor Interface Module) library interfaces a wide range of sensors with ROS over the Arduino microcontroller, but only the Devantech TPA81 Thermal Array and the LSE Thermistor Anemometer are relevant here (ROS ardusim, rosserial_arduino). The nxt_ros package creates a ROS topic for each of NXT sensors and publishes the sensor's data on this topic. Only the color sensor could be relevant (ROS nxt_ros). Phidgets sensors can be interfaced to ROS via phidgets_ros package (ROS phidgets_ros), but their assortment is poor and they are suited rather for the laboratory environment. Sensoray626 package provides a ROS wrapper for the Sensoray Model 626 analog and digital I/O card drivers, working in over zero temperatures only (ROS sensoray626). Finally sr_ronex package interfaces to the Shadow RoNeX, a high speed (100Mbps) interface bus for ROS built on robust industrial standards. It connects motors, sensors and other peripherals via an ethernet port and accesses each directly using ROS and RoNeX's open source drivers, but is not operational outside the 0 – 55ºC temperature domain, much too conservative for the real CFR (oil-and-gas) applications (ROS sr_ronex).

Fine maneuvering

A specific requirement in the (non-visual) inspection domain may be the requirement of the fine positioning (fine maneuvering). Inspection of the industrial equipment requires accurate positioning to take measurements (within the „cone of sensor vision”), and possible re-positioning depending on the measurements already taken (e.g. when sensing gas leakage or flame plume after having checked the local wind direction).

The required location and pose for the inspection measurements belongs to the Inspection Plan description, is available in the KDB and can be used to set the final configuration to compute the movement trajectory to access the equipment. When the CFR robot negotiates a longer segment of the access road however, its final location and pose close to the inspected equipment may differ and may require corrections.

The problem with the correction is that the normal trajectory computation done by the local planner in the ROS navigation stack prefers the movement ahead; in the direction of sight of the obstacle sensing sensors (see Section 4.2.4)[8]. If the correction of the location and/or pose would require a slight movement aside, or aback, as it is practical in the small space close to the equipment, the conservative trajectory computation will produce excessive movements. Further simplifying assumptions are that when positioning for the measurements already in the vicinity of the equipment, no static or dynamic (human) obstacles are expected, and the obstacle avoidance mechanism with the obstacle inflation may be omitted (handling obstacles belongs to the access trajectory management, but once the CFR arrives at the equipment access point we can safely assume that the potential obstacles have been cleared).

To solve the fine correction problem we suggest after (Leibold 2012) to use a fine maneuvering navigation unit overriding when needed the navigation stack, see Fig 13.

[8] This is the situation with the basics planners recommended in the packages. There are also local planners admitting the backward movements, like e.g. TED (Timed Elastic Band) planner, and it is up to the CFR designer which mechanism to use for the fine maneuvering in the proximity of the inspected equipment. Nevertheless an architectural option for an independent maneuvering in place (as opposed to the move-ahead trajectory control) seems advisable.

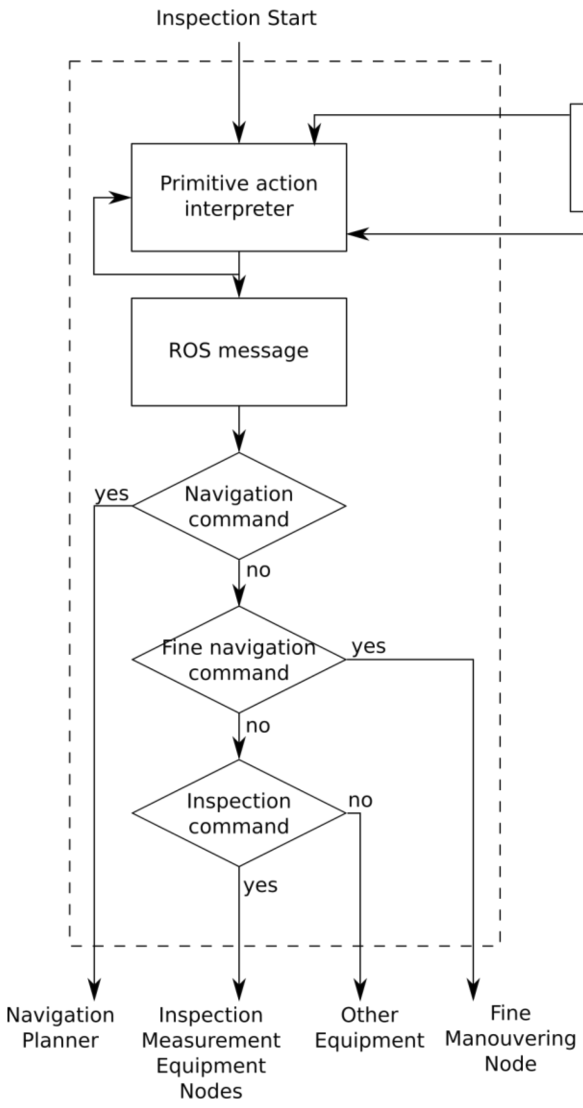


Fig 9. The control flow of the Inspection Planner.

4.2.3 Mission and installation representation

The modules of the Planning and Reasoning Architecture (in particular the Mission Control and the Inspection Planner) need to explicitly represent and store in a knowledge base the data and information required to perform the planning and replanning operations and the execution of the ready plans. This component is the Knowledge and Database (KDB) containing the static and run-time data provided by the operators before the mission, computed or collected during the mission, and summarized and qualified after the mission. Accordingly to the current design decisions the KDB will be implemented as a CFR onboard database server, based on PostgreSQL, as it offers a good balance between functionality and system requirements (PostgreSQL). The details of the architecture (data model) and the information content of the KDB will be finalized and fully described in the next deliverable (R5-COP D26.22) by M33 deadline.

In the following we give a short synopsis of the data structure of the information stored in the KDB relevant to the Planning and Reasoning Architecture. The details of the representation and the data structured will be described in the companion deliverable D26.22 Knowledge and Database.

- (**Inspection**) **Mission**: contains references to the CFR robot, used mission plans, description of the installation, and the reports of the concluded missions.
- **Installation**: refers to the site documentation, semantic maps describing the lay-out of the installation and the access roads, and the structural description of the installation process modules listing their equipment (equipment inspection points).
- **Equipment**: specifies the kind of the measurand to be inspected, the location and pose of the CFR to perform the inspection, the accuracy and the validity limits for the measured value, and the Inspection Plan Stub relevant for this equipment.
- **CFR**: is the operational description of the co-worker robot, its actual status included (what is faulty).
- **CFR configuration**: is the technical description of the mission capabilities and limitations of the robot, deployed sensors, actuators, and other hardware, the capacity of the battery, the way how to communicate with the robot, etc.
- **Sensor**: is basically the simplified sensor data sheet, with added references to the ROS packages encapsulating the sensor function.
- **Actuator**: same for the actuator hardware.
- **Measurand**: is the physical or chemical quantity to be inspected at a given equipment inspection point together with a list of sensors capable of sensing this kind of quantity.
- **Plans**: stores every kind of plan related information. Prior to the mission it receives the Functional Mission Plan (i.e. the requirements and constraints on the inspected equipment) and (in the basic version) the manually precomputed Inspection Plan Stubs for every equipment demanding inspection. Then in the due course of the mission planning and executing Plans fills in with computed more detailed plans, the optimized Mission Plan, and the optimal Metric Mission Plan adjusted to the lay-out of the access roads. If due to the operator's considerations or the experienced run-time difficulties the original plan must be pruned and re-computed, these plans will be also loaded here beside the original ones. (Plans would be also the storage place for the Inspection HTN plans, see Section 5.2, if the Inspection Plan Stubs would be computed automatically with a HTN planner).
- **ROS package**: descriptor presents the consisted information about ROS packages. It lists the administrative and versioning data, dependencies upon other packages, the package ROS interface (topics, messages), the functional capabilities represented by the package's nodes, and the hardware encapsulated by the package. An important kind of packages will be plug-in algorithm and procedure libraries (see Navigation Planner).
- **Report**: collects the mission data, measurement results at the inspection point, their qualification, and every event relevant to the interpretation of the mission (the unexpected, the communication with the operator and his decisions, the status of the equipment and the robot, the environmental conditions, etc.)

- **Manual**: is the repository of the documents considered helpful, to be recalled on the CFR GUI (CFR manuals, measurement manuals, installation related protocols, …).
- **Maps**: is the map server providing any kind of (static) semantic and binary map describing the lay-out of the installation, lay-out of the access roads, geometric shapes of specific areas, and features of any object expected to be found during the inspection. As the representation tool Keyhole Markup Language (KML) was chosen - a much extended XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers. KML representation will allow referring to the inspection points and other relevant locations by human-understandable names instead of GPS or other reference frames coordinates.

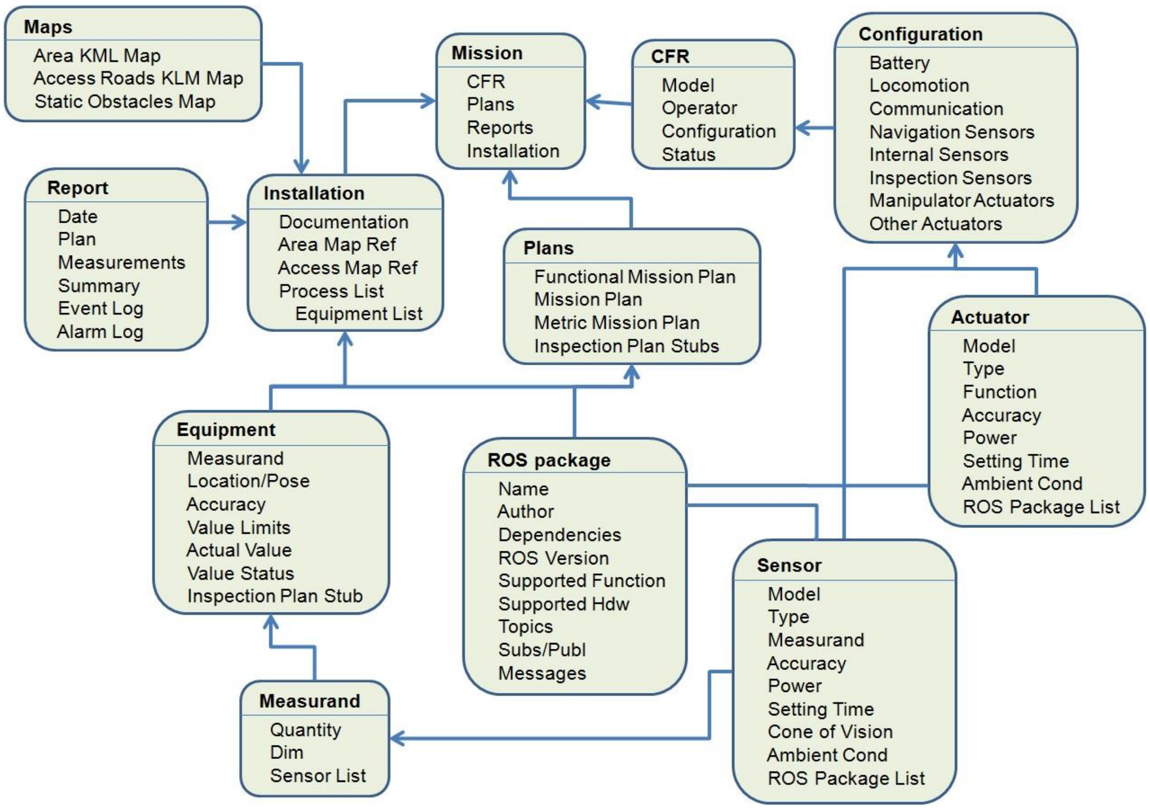


Fig 10. Schema of the KDB items relevant to the computation of the Metric Mission Plan and the Inspection Plans (the arrows represent the reference hierarchy).

4.2.4 From the Inspection Plan to the Navigation

The backbone of the CFR inspection tour is to navigate along the industrial installation access roads. It is the task for the Navigation Planner and a standard task in the mobile robotics, for which plenty of resources are already available. One of them is the ROS Navigation Stack, a package providing global and local movement planning and executing services.

Standard ROS Navigation Stack

ROS Navigation Stack is a 2D navigation stack that takes in information from odometry, sensors, and a goal pose and outputs safe velocity commands that are sent to a mobile base (ROS Navigation Stack, Mobile Base). The proper functioning of the Navigation Stack demands a differential drive, a planar laser (or equivalent sensors, Kinect), and its performance is best for nearly square (circular) robots. All these requirements are fulfilled in case of the CFR prototyped on the Summit XL HL mobile robot platform from Robotnik (Robotnik).

To produce safe velocity commands the Navigation Stack must be provided with map and localization information, which will be then re-worked by planners into the trajectory and velocity information. The usual way of the environment mapping is the Simultaneous Localization and Mapping (SLAM) technique which builds up a map within an unknown environment while keeping track of the positions.

In case of the CFR the structure of the navigable environment is partly known (static maps in the KDB with the location of the access roads and the equipment sites). This information must be up-dated to the full 2D occupancy grid map with the real-time sensor data (laser scans, Kinect data, etc.) and the odometry. Static information can help to identify the movement corridors, and the non-identified occupancy means unexpected obstacles. The occupancy grid map (with defined resolution, origin, and occupied/free grid value thresholds) is published by the map_server under the /map topic (ROS map_server).

Next is the localization which means estimating the pose of the robot relative to a map. In the standard Navigation Stack the localization is solved with the amcl package (Adaptive Monte Carlo Localization) (ROS amcl). AMCL is a probabilistic localization system for a robot moving in 2D, which uses a particle filter to track the pose of a robot against a known map (in current implementation it works only with laser scans and laser maps). A more involved localization mechanism based on sensor fusion governed by Kalman-filters is the Extended Kalman Filter (EKF) localization (ROS robot_localization).

The move_base package computes the controls to move a robot to the desired position (ROS move_base). The move_base node links together a global and local planner to accomplish the global navigation task, but may optionally perform trajectory recoveries when the robot is stuck among obstacles.

The move_base supports any global planner adhering to the nav_core::BaseGlobalPlanner interface and any local planner adhering to the nav_core::BaseLocalPlanner interfaces specified in the nav_core package (ROS nav_core). It maintains also two costmaps, one for the global planner, and one for a local planner (ROS costmap_2d). The costmap represents places that are safe for the robot to be in. The values in the costmap are usually binary, representing free space or the danger of the collision. The global costmap is used for the global navigation, i.e. to create paths to far-off distant goal places. The local costmap covers smaller area and is used for the local navigation, to compute paths to the close destinations and to avoid obstacles. Both maps are periodically updated and published.

The full scale (global) navigation is realized by the linked global and local planners, using acc. the global and the local maps. The global planner can be realized e.g. with the navfn package (ROS navfn), and the local planner with the base_local_planner package (ROS base_local_planner). The global planner uses the (global) costmap to find a minimum cost plan from a start point to an end point in a grid and computes the global plan before the robot starts moving toward the next destination. In the current implementation (ROS navfn) the used algorithm is Dijkstra's algorithm. A* algorithm is supported in (ROS global_planner) planner (see also foot-note in Section 4.2.1). Every time the planner computes a new path, the last plan computed is published on the topic /move_base_node/NavfnROS/plan.

The local planner uses the incoming sensor data to choose appropriate linear and angular velocities for the robot to traverse the current segment of the global path. The base_local_planner combines odometry data with both global and local costmaps to select a kinematic trajectory for the robot to follow. As the generic planning methods Trajectory Rollout and the Dynamic Window Approach (DWA) are implemented (ROS dwa_local_planner) (Fox 1997). Both differ in how the robot's control space is sampled, but are reported to perform comparably in tests, with the recommendation to use of DWA for its efficiency gains.

The local planning searches for a suitable local plan generating a number of candidate trajectories in every control cycle. Every generated trajectory is checked whether it collides with an obstacle. If not, a score is given to compare trajectories and to pick the best. To other ROS nodes the local planner publishes the topics of /global_plan (part of the global plan corresponding to the local trajectory), /local_plan, and the /cost_cloud used in scoring.

The base_local_planner package is equipped with a number of cost functions, scoring the (obstacle avoiding) trajectory depending on how closely the trajectory follows a global path. Special scoring punishes also the oscillations and prefers motions in the front direction, penalizing backwards and strafing motions.

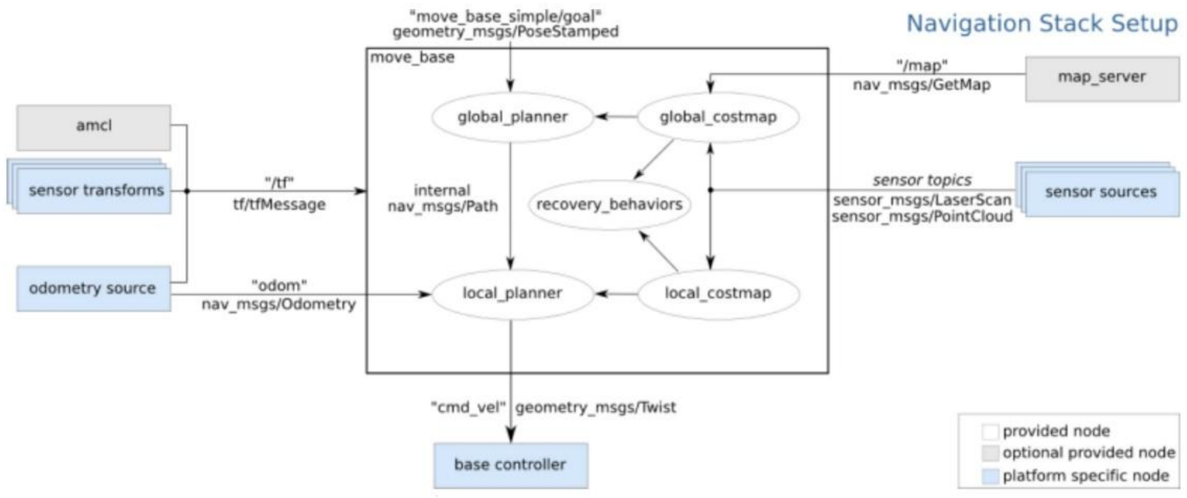


Fig 11. Architecture of the ROS Navigation Stack (from http://wiki.ros.org/navigation/Tutorials/RobotSetup)

The ROS Navigation Stack, no matter how rich in functions, won't be enough for navigating in the industrial inspection domain. In the following we consider some functional add-ons to the Navigation Stack, making it into a fully flexible Navigation Planner.

Fine maneuvering unit

It was already mentioned that the (non-visual) inspection domain may require fine positioning (fine maneuvering into a position) to make monitoring observations of the demanded quality.

Normal trajectory computation implemented in the Navigation Stack prefers the movement ahead, in the direction of the cone of vision of the obstacle seeking sensor (see PreferForwardCostFunction in (ROS base_local_planner). This is a sound strategy when the obstacle avoidance is at stake and moving into an un-explored direction could be dangerous. However in the immediate vicinity of the inspection point we may freely assume that no static or dynamic obstacles are present (otherwise they would have been observed from ahead and handled accordingly). So the obstacle avoidance mechanism may be switched off and small corrective actions like moving aside, or aback, are permissible and practical.

It is plainly reported in the ROS package description that the preference for the ahead direction (and penalizing backwards and strafing motion) was designed with robots like the PR2, having sensor coverage only in front. It is also noted that on other robots or in other domains this may not be a pursuable solution (ROS base_local_planner). To solve the problem a fine maneuvering navigation unit is introduced in Fig 13, to override when needed the basic Navigation Stack commands (see also (Leibold 2012)).

Planners plug-ins

Planners used in the Navigation Stack must adhere (must be wrapped) to the specification nav_core::BaseGlobalPlanner, nav_core::BaseLocalPlanner, nav_core::RecoveryBehavior, defined in the nav_core package (ROS nav_core). That way the robotic system designer can exchange the built-in options of the planner, local controller, or recovery behavior for new versions (as plug-ins) adhering to the same interface.

The current global planners using the nav_core::BaseGlobalPlanner interface are:

- The global_planner package (ROS global_planner), using Dijkstra or A* search algorithms,
- Grid-based global planner package (ROS navfn), assumes a circular robot and navigates with Dijkstra algorithm,
- So called carrot_planner (ROS carrot_planner) - a simple planner that attempts to move the robot as close as possible to the goal, even when the goal is within an obstacle.

The current local planners using the nav_core::BaseLocalPlanner interface are:

- The base_local_planner (ROS base_local_planner), which implements the Dynamic Window and Trajectory Rollout methods,
- The eband_local_planner (ROS eband_local_planner), which implements the Elastic Band method, and
- The teb_local_planner (ROS teb_local_planner), which implements the Timed-Elastic-Band method for online trajectory optimization.

The nav_core provides also standard (costmap) recovery behaviors: reverting the costmaps used by move_base to the static map outside of a user-specified range, and 360 degree rotating of the robot to attempt to clear out space.

If these choices of the planners and the recovery mechanisms are not satisfactory for a given robot configuration or application domain, then alternative algorithms can be designed and encapsulated in the plug-in algorithm library pluginlib (ROS pluginlib), which is a C++ library for loading and unloading plug-ins from within a ROS package (see also (Tomović 2012)).

The plug-in providers are required to register their plug-ins in the package.xml of their package and to describe the plug-in in a description file. Actually tutorials are provided guiding the prospective designers to plug-in other global path planner algorithms (ROS Adding Relaxed Astar, Global Planner As Plugin).

The plug-in option is of particular importance to the Planning and Reasoning Architecture in the co-worker domain, considering that in this case more involved optimal path and action planning algorithms are required (see Sections 3.1.1, 5.2). A portable, ROS-immersed implementation of advanced inspection planners would be an enormous asset to the robotic community.

Velocity command multiplexing

In the Navigation Planner, in principle, four independent components may issue velocity commands competing for the mobile base (Navigation Stack, teleoperation, fine positioning, emergency stop, and the optional stand-alone obstacle avoidance). The multiple velocity commands realize different priority goals and differ in the publishing frequency and the magnitude of the control signals. In the worst case many different velocities may be issued simultaneously to the motor making the control chaotic.

ROS contains a tool to filter and selectively apply velocity commands. It is a command multiplexer which arbitrates the incoming cmd_vel messages from several topics and allows one topic at a time to command the robot, based on the priorities (ROS cmd_vel_mux, yocs_cmd_vel_mux). Command multiplexer also deallocates current allowed topic if no messages are received after a configured timeout. All the motor control topics, their priority and timeouts can be re-configured at runtime.

An additional velocity control can be provided by the velocity smoother (ROS yocs_velocity_smoother) which subscribes and publishes the same velocity message topic, but limits the control content of the incoming velocity message accordingly to the velocity and acceleration limits of the robot.

Layers and scoring plug-ins

The basic mechanism of the static and dynamic obstacle avoidance in the Navigation Stack is the layered construction of the local costmap, where the dynamic obstacles are copied over the fragment of the static map, obstacle boundaries are inflated (to account for the no-zero robot dimensions) and the grid cells on the resulting map are numerically scored (Static Map Layer, Obstacle Map Layer, Inflation Layer, see (ROS base_local_planner, costmap_2d). The obstacle and voxel layers automatically subscribe to sensors (point clouds or laser scans), which continuously insert and remove obstacles from the layer. The local planner seeks then a trajectory along the least scoring grid cells (Cell scoring: No information 255, Obstacle 254, Inflated obstacle 253, Free space 0, with an exponential fall-off in between) (ROS costmap_2d, costmap_2d/hydro/inflation). Other layers can also be implemented and used in the costmap via pluginlib option.

This standard layering and scoring leaves much to amend in the demanding domain of the CFR inspection tours. The movement environment of the CFR is structured, there are no free country dirt roads, various natural obstacles, only well defined (and perhaps well tended) access roads with concrete lanes (probably with no negative obstacles). The access roads are mainly straight, relatively narrow segments with crossings.

This good news is countered by the human traffic on the roads. Traffic can be pedestrians, bicyclists, and even people driving carts or trolleys. The human traffic belongs to the industrial installation in a sense that people on the roads are professionals, move purposefully, observe the rules of the road, and move relatively fast, without strolling. They expect not to be distracted in their goals, but exactly the same applies to the CFR, negotiating the common access roads also on purpose.

Luckily meeting people on the road may be limited to a small number of traffic situations. The CFR may move along a hallway, a linear segment of the road. No problem if the road is empty (within the cone of vision of the obstacle sensors), if there is traffic coming in the opposite lane and/or if there is traffic ahead of the CFR but moving faster. The CFR just continues its mission. The situation becomes more involved if the traffic ahead is slower, blocking the CFR in progression. Such dynamic obstacle may be overtaken, observing the comfortable personal space around it, not cutting in immediately etc. (see Appendix A) The situation becomes even more complicated if there is a noticeable traffic coming in the opposite lane, e.g. a fast bicycle (see Appendix A for a draft computation which may serve as a basis of the suitable trajectory scoring procedure). Now a tough decision must be made: to overtake fast without frightening anybody, or to slow down and to wait out the momentary congestion. Another similar well structured situation is when the CFR is closing to a crossing.

To the specifics of the CFR domain belongs that the obstacle interpretation depends on the working regime of the robot. During normal mission the dynamic human obstacles must be maximally respected, but in emergency it is the CFR who has the priority of the road and pushes through, helping itself with warning lights and sounds.

Due to the narrow access road passages specific static non-permanent obstacles are the products of heavy weather: snow mounds, pools of water, layers of hail. A specific problem is also how to effectively recognize a bicyclist coming from ahead, when the silhouette will be the least representative, and when it is important to judge well the velocity (Cho 2010, Buisson 2013).

In the light of the increasing number of data sources with an individual semantic meaning influencing the trajectory scoring, the monolithic 3 layers costmap with binary values (free space vs. lethal obstacle) and an integral scoring does not seem appropriate, and we should turn to the plug-in options instead introducing more layers with much differentiated scoring.

After (Lu 2014) we propose multiple layered costmaps (fitting well into the idea of the plug-in, Fig 12). In the layered costmap system every layer can be up-dated and scored separately, tracking situations of a specific functionality. The layers are then fused together to give the final accumulated scoring for the trajectory selection. (Lu 2014) lists numerous benefits of such approach: much clearer map updating, selective and dynamic (flexible configurations) updating, and the option for semantically meaningful ordering of the score updating. Once the pool of the obstacles can be better differentiated, we can make better service from various forms of collision and proximity detection algorithms (see e.g. (Pan 2012)).

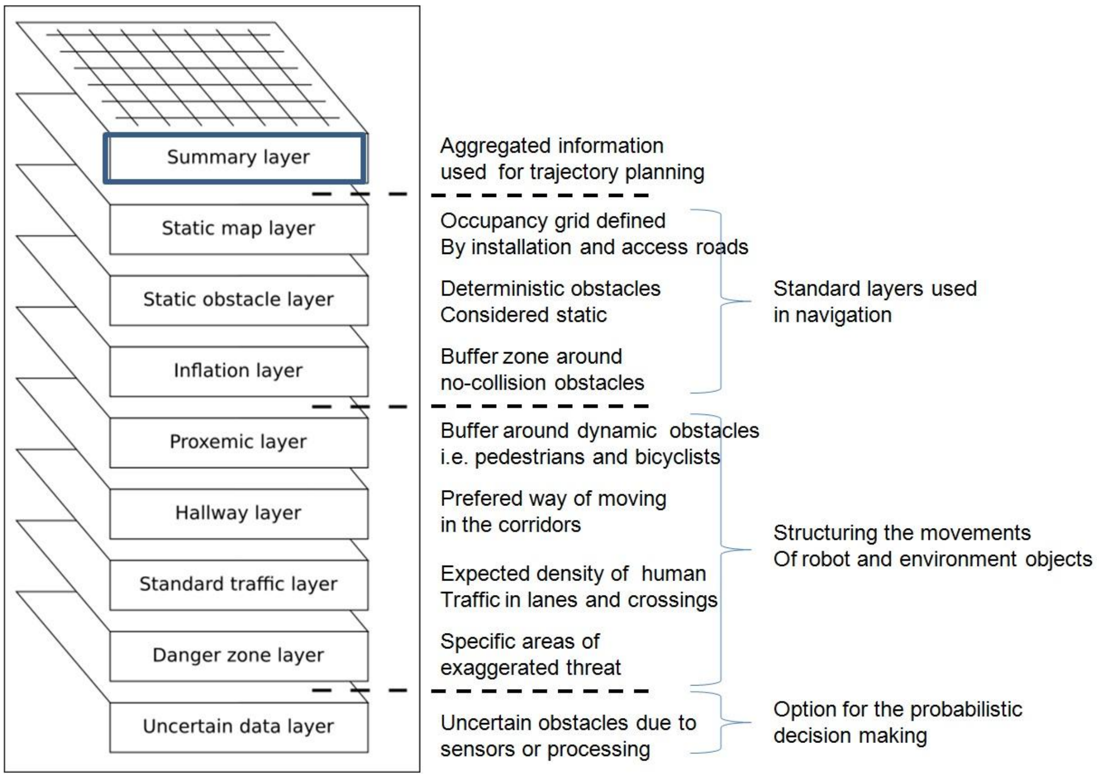

Summary layer
Static map layer
Static obstacle layer
Inflation layer
Proxemic layer
Hallway layer
Standard traffic layer
Danger zone layer
Uncertain data layer
Aggregated information used for trajectory planning
Occupancy grid defined By installation and access roads
Deterministic obstacles Considered static
Buffer zone around no-collision obstacles
Standard layers used in navigation
Buffer around dynamic obstacles i.e. pedestrians and bicyclists
Prefered way of moving in the corridors
Expected density of human Traffic in lanes and crossings
Specific areas of exaggerated threat
Structuring the movements Of robot and environment objects
Uncertain obstacles due to sensors or processing
Option for the probabilistic decision making

Fig 12. Proposed map layering for the CFR trajectory scoring (after (Lu 2014)).

Obstacle avoiding with learning (option)

The highly structured environment makes it theoretically possible to learn the behaviour of the local traffic around the CFR and to teach it how to response to it.

An interesting option based on the velocity multiplexer and getting around the need to explicitly define and score obstacles is to design a neural network with the velocity output and to teach it on the human traffic data with input features like distance, direction (velocity vector), perhaps some of the environmental ambient conditions (fog, rain, wind, …), the part of the day, the specific location, etc. (see Fig 13) (Martínez-Gómez 2014). A more involved learning structure, fusing the decisions of a number of more specialized (to the specific situations) neural networks (Mixture of Experts – ME) can also be tried (Jordan 1993).

The success of the proposed solution depends on the quality of learning and this depends primarily on the quality of the learning and test examples. Considering that the CFR domain is too difficult to be modeled theoretically, the examples can be gained from the experience (after the prototype of the CFR has been designed and employed), or from a carefully set up and parameterized simulation.

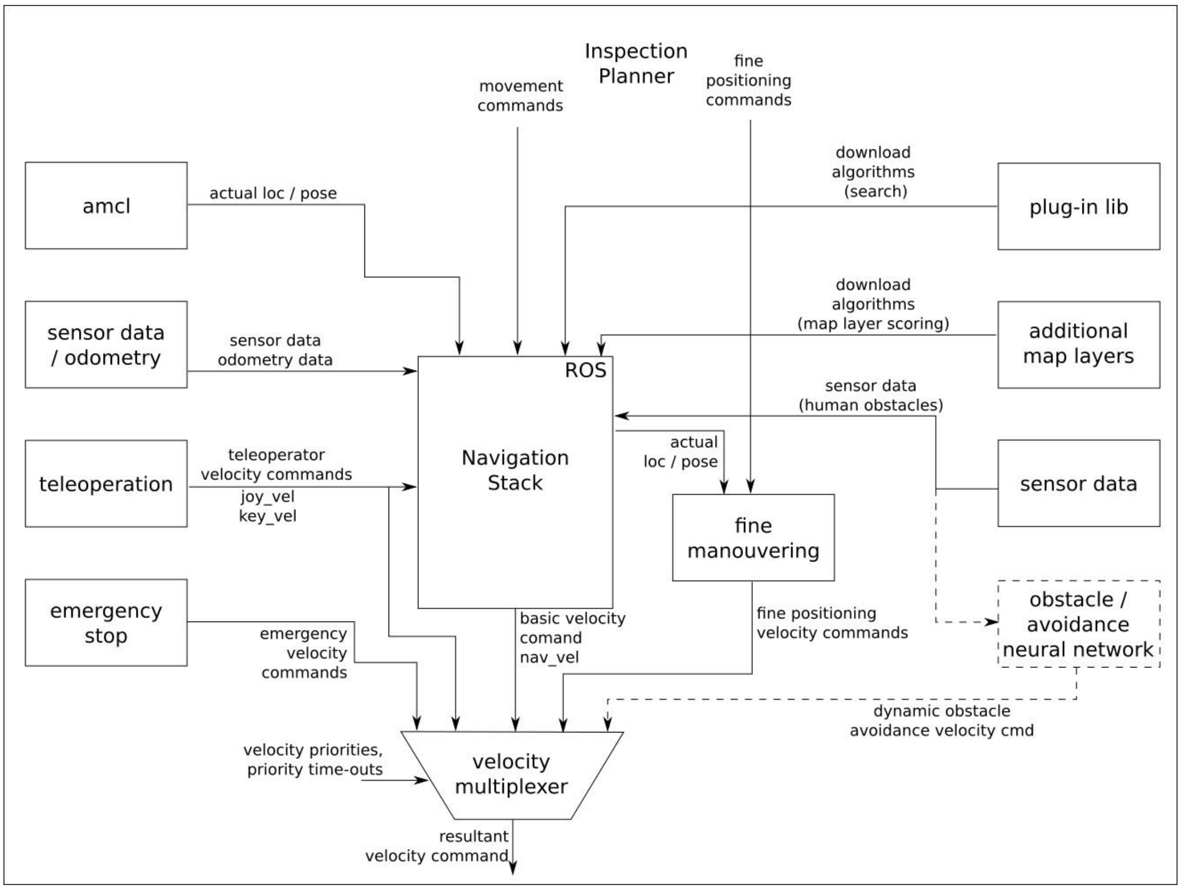

Inspection Planner
movement commands
fine positioning commands
amcl
actual loc / pose
download algorithms (search)
plug-in lib
sensor data / odometry
sensor data odometry data
download algorithms (map layer scoring)
additional map layers
ROS
Navigation Stack
sensor data (human obstacles)
teleoperation
teleoperator velocity commands
joy_vel
key_vel
actual loc / pose
sensor data
fine manouvering
emergency stop
emergency velocity commands
basic velocity comand nav_vel
fine positioning velocity commands
obstacle / avoidance neural network
dynamic obstacle avoidance velocity cmd
velocity priorities, priority time-outs
velocity multiplexer
resultant velocity command

Fig 13. Architecture of the Navigation Planner with an embedded ROS Navigation Stack. The broken line part indicates the optional obstacle avoidance learning.

4.3 Reasoning about plans – scenarios for (re)planning

The presented architecture is responsible for the control flow of the planning (how the plans are designed and fused together) and replanning, when sensory information comes in. In the following some typical use cases are reviewed involving reasoning about planning, tracing the activity of the components and the interaction between the CFR and the staff.

Plans computed by different planners must be properly integrated and executed. Integration of the mission and the inspection plans is relatively straightforward by „cementing” the path segments of the mission plan with the inspection plans during execution. Path segments from the mission plan are then fetched one after the other, decoded to the true road segments and passed to the movement planning and avoidance control. After the current path segment had been successfully negotiated the following inspection plan is fetched and passed to the peripheral (sensor and actuator) and movement control. When moving along the path the movement control executes the human traffic avoidance strategies.

1. Default inspection

It is the case when the map of the facility and every functional information is available prior to the inspection and a full set of mission/inspection plans can be prepared in advance (OpC Command Module, or Mission Control) and other control parameters (e.g. human avoidance strategy) chosen. The plans are then passed over to the CFR Inspection and Navigation Planners for execution.

Fail/Replan: If the computed plan requires more battery than it is available, iterative pruning of the inspection plans is needed. If no satisfactory plan reduction is possible, then the planner could always plan for an intermediate battery replacement/recharge, or finally some re-design of the robot power supply may be in the offing.

2. Nonpassable obstacles recognized prior to the mission

If the site operator or the CFR operator receives a notion about an obstacle being permanent in time of routing the mission, he can re-schedule the mission to an earlier or later time span, keeping the original plans.

Fail/Replan: If the mission re-scheduling is not possible, the obstacle must be introduced to the access path graph (by amending the static semantic map), leading to the computation of the new mission plan (as in 1.) to be downloaded to the CFR for execution.

Considering that the original plan was battery optimal, the new plan may be heavy on the battery. Additional pruning of inspection tasks may be needed.

If no satisfactory plan reduction is possible, CFR must suspend and the human staff must be sent for inspection.

After the obstacle has been managed (physically eliminated and deleted from the map), return to the original plan.

3. Nonpassable obstacles recognized during the mission (by operator)

If the site operator or the CFR operator does receive a notion about an obstacle happening in the time of the inspection route, he must check how serious the problem is by pruning the access graph due to the obstacle and by recomputing the mission plan from the actual inspection point (by pruning in the Functional Mission Plan the equipment already inspected). If no change with respect to the original plan is demanded, the CFR can continue.

Note: If the obstacle is far from the actual place of the robot, it can continue the mission in the mean time. If not, the robot should be stopped until the calculation is done.

Fail/Replan: A new computed partial plan is checked for battery life and passed to the robot for execution. The new plan may be heavy on battery (additional by-pass roads). Additional pruning of tasks may be needed, besides the path pruning. If no satisfactory plan reduction can be achieved, the CFR must be suspended and the human staff sent to clear the obstacle.

4. Nonpassable obstacles recognized during the inspection (by robot)

If on the autonomous mission the robot observes a nonpassable obstacle, it must stop and notify the operator. The operator then continues as in case 3.

5. Emergency

If on the autonomous mission the robot observes an alarming phenomenon, it must stop and notify the operator. The operator takes over by teleoperating, commands the robot to continue the mission, or the robot may be ordered to a remote location. It should take the fastest (shortest) route to the destination, driving with warning lights and sounds.

The alarm route should be planned with A* search (or some of its suboptimal version, keeping in mind that in case of the alarm it is the overall plan computation time + the plan execution time, what counts) run on the true access road graph. When computing the edge costs:

- The costs of the inspections should be deleted (no inspection activity on the way),

- The maneuvering costs of the human obstacle avoidance should be decreased (robot runs with alarm lights on, horn blaring, the human should adapt and actively avoid),

- The uniformly movement costs should be increased (higher speed, higher power demands).

The replanning can be made on board CFR for the autonomy, or on the operator system for the subsequent plan-supported teleoperation.

Fail/Replan: In case of obstacle, the control should be passed to the operator for a fully manual teleoperation (or any other decision).

6. Inspection with a faulty CFR (fault known prior to the inspection)

Assume that some of the inspection functions of the CFR are not operational, cannot be repaired in time, yet the remaining functions (locomotion and obstacle avoidance included) can provide meaningful inspection data. The original plan must be reviewed by the operator (at the center, or on-board) and the non-operational actions removed:

(a) Removing blocks in the Inspection Plan Stubs related to the non-operational sensors,

(b) Removing entire equipment inspection points in the Functional Mission Plan, if due to the faulty sensors no measurements can be done at all.

The reduced plan, case (a), can be managed by the Inspection Planner in the run-time, deleting from the processing the non-functional blocks. Case (b) leads to the replanning of the whole mission, must be done under the supervision of the operator and then loaded to the CFR for execution.

Fail/Replan: The reduced capabilities and plan may be insufficient for the emergency purposes (essential alarming phenomena cannot be observed). This problem (go/no go) must be decided by the operator.

7. Inspection with a faulty CFR (fault happening during the inspection)

Assume that some of the inspection functions of the CFR become non-operational, and the CFR is able to detect it and to interpret it correctly (self-test, calibration, etc.). The inspection must be paused, the operator notified. Upon the operator decision the CFR must return to the home shelter, or must refresh its plan by deleting the non-functional actions, and continue. The plan refreshment is done by the Inspection Planner skipping the control of the faulty equipment.

Fail/Replan: The option 6b optimizing the mission by evading equipment sites which cannot be now inspected is too difficult to be computed during the active mission. It may also be possible that the sole visual inspection opportunity is valuable to the operator and the original mission route should be kept.

8. Withdrawing to the shelter

When, due to some reasons, the inspection route must be discontinued, however with no emergency status, the CFR should take the route to the shelter destination. The route can be the fastest (the shortest), but also different policies could be tried (e.g. a route involving less contact with the human traffic). The return route should be planned with A* search run on the true access road graph. Contrary to the emergency, only the costs of the inspection should be deleted (no inspection activity upon return). The replanning can be made on board CFR for the autonomy, or on the operator system for downloading or teleoperation.

9. „Continue from here”

It may happen that a part of the plan must be cancelled because a part of the installation becomes non-passable or non-accessible (like the obstacle detected in the earlier use cases), but the operator is aware (upon evaluating the plan vs. the situation) that the inspection can be continued from a location further on, with a safe return to the shelter, without the need for replanning. The operator brings the CFR into the teleoperated regime, takes it to the picked location and orders to advance the mission plan to that location as the starting point and continue.

5 Conclusion and further developments

5.1 About the Planning and Reasoning Architecture on the whole

Is the proposed Planning and Reasoning Architecture sound?

Given an application domain and a problem to solve, abstract analysis of the goals yields the formulation of the tasks. The tasks in turn lead to plans in terms of even smaller tasks, etc., until the level of directly applicable actions is reached, where the abstraction give way to the direct reference to the robot kinematics and dynamics.

This wide gap between the goal/task level (symbolic planning, task planning, and involving symbolic reasoning) and the object manipulation, negotiating rough terrain and avoiding obstacles (motion planning, involving geometrical reasoning) can be bridged in a number of ways.

(1) Traditional approach is to treat them separately, with an explicit hierarchical decomposition of the planning problems. Different levels of abstraction usually call for different formal representations, and those for different mechanisms of the planners. Problem formulated at a given abstraction level is solved (by planning) at that plan becomes a back-bone to formulate and solve the problems lower on.

The net gain is that abstract problems are structuraly simpler and cheaper to solve, but their solutions are not applicable in practice. Tackling problem at the low level fully amplifies the complexity and makes its solution difficult.

The gradual hierarchical decomposition helps to overcome these difficulties, because the abstract plan backbone serves as the useful control heuristic when running more detailed problem solvers. So (see e.g. the services of the Navigation Stack in ROS) the optimal global movement route is solved with the Astar global planner using path segment steps as the action primitives, then these segments are sequenced as subgoals to the local planner, taking into account the robot movement capabilities and its immediate surroundings, producing as the action primitives direct commands to wheels or grippers.

The clear drawback of the hierarchical approach is that the resulting robot plan will never be fully optimal. High level plan computes solution without taking into account how the robot really can drive and what will it find on the road. Low level computations conform in turn.

(2) An alternative to the hierarchical decomposition is to consider the level of the executable elementary actions (commands) and to provide the solution to the abstract goals expressed as elementary action plans. The complexity of such approach will usually be prohibitely high.

There are multiple, sometimes very ingenious indeed, approaches attempting to integrate the task and the movement planning within a single representation and planner.

One track is to introduce geometrical reasoning checking the feasibility of the symbolic operators when they are picked to be inserted into a task level plan (Cambon 2004ab, 2009, de Silva 2013). Such plan can be then translated into the executable primitives and executed without problems. Another approach is to formulate the task level goals in the robot configuration space (location and pose) and to solve it for the optimal path using traditional search based planners (e.g. A*) (Arain 2015, Ulrich 2016). Yet another approach is search-based planning using motion promitives of the robot, i.e. in a given search state/node the node expansion happens accordingly to a finite set of robot moves precomputed for each robot orientation, observing the kinematic constraints and filtered by the presence of the obstacles (Likhachev 2010, ROS sbpl, Petereit 2012). The search itself is one of the usual optimal heuristic algorithms.

In case of the co-worker robot, not only moving around, but performing also inspections, the problem is more involved. Inspection can be "homogeneous" in a sense that the robot must sample (moving around) its whole 2D/3D environment, and at a number of places (inspection points) it must perform observations of the same character (Arain 2015b, Cacace 2014, Englot 2011, 2012, Faigl 2011, Galceran 2013). Such is the perimeter or area guarding problem, pollution mapping problem, 3D structure integrity checking problem (in air, or under the water). "Homogeneous" inspection can be treated by a flat, non-hierarchical planning.

Inspection duties addressed in the present deliverable are of different, "heterogenous" character. The optimum problem can be formulated meaningfully only at the task level. At the lowest level of direct movement the actual situation can be so unpredictable, that the only feasible approach is the reactive planning, governed "from the above" by the routing subgoals issued from the task level (Buisson 2013, Cho 2010, de Carvalho 2013).

The real challenge lies in between. At least a part of the local inspection duties means quite different, non-movement kind of activities, possibly ruled by specific domain and application dependent protocols (Arain 2015a, Barber 2015, Habib 2014, Ibarguren 2013, Liu 2012, Maurtuaa 2014, Reggente 2009, Soldan 2012, TALON 2004, Sensabot 2012, Spiewak 2015). Such duties must be also planned and executed, but they are too local for the global route optimization and independent from the short distance movements.

In conclusion we can volunteer the opinion that the proposed modular Planning and Reasoning Architecture reflects well the hierarchical structure of the industrial inspection problem and that the loose coupling among the modules facilitates the introduction of different planners best fitting the particular phases of the planning problem.

5.2 Planning the inspection measurements

Inspection activities (plan stub) at a particular given inspection point may cover diverse measurements, status and value checks, ask-and-acknowledge communication with the operator, local maneuvering, repeated and conditional activities, derived computation, result storage, etc. All these activities are functionally and perhaps hierarchically related (they happen in the vicinity of a particular hardware), possibly partially ordered, and dependent on some free parameters.

Formal planning methods well suited and routinely usable to the planning of the inspection (measurement, maintenance) activities are as yet nonexistent. The reported co-worker inspection robots are equipped with the measurement hardware and presented as able to do measurements, however no substantial information is provided how the domain specific inspection protocols were translated into the sensor-level control commands (Arain 2015a, Habib 2014, Ibarguren 2013, Maurtuaa 2014, Reggente 2009, Soldan 2012, TALON 2004, Sensabot 2012, Spiewak 2015). They were probably teleoperated or programmed for the measurements manually.

In the field of generative task planning, spanning the gap from the inspection goal to the executable inspection commands two attempts are worth mentioning.

A mobile robot with an 6 DOF arm and a gripper was programmed for ssisting handapped/ ageing persons in their apartment. The used representation was the PDDL (PDDL), the planner CPT (optimal temporal POCL planner based on constraints), and the low level commands were represented as an event-based finite state automata linking high level states to executable functions controlling the robot (Morignot 2010, 2011, SAM).

Another attempt is the ROSPlan, a task planner coupled to the ROS level commands (Cashmore 2015), presenting the case study involving autonomous underwater vehicles maintaining walves in a temporary constrained manner. In the ROSPlan its Planning System builds the initial state automatically (as a PDDL 2.1 problem instance), passes it to the planner, processes and validates the plan, and dispatches the plan actions for execution as ROS action messages. The paper presents a planning example, but does not elaborate how the PDDL action –to-ROS message translation is realized formally, especially if the hardware, the ROS packages, the topics and messages would change without any modification to the goals at the task level.

Some critics also apply to the chosen task level representation of the PDDL 2.1. This version of the PDDL language was introduced in (Fox 2003) and currently PDDL 3.1 (PDDL) is used in planning competitions. Already in 2003 serious critica were formulated against the language definition. It was pointed out that a limited notion of durative actions incorporated into the language was clumsy and the process of decomposing an action into sub-actions was very complex. The reason was traced back to the basic property of the generative planning that we should not be able to directly specify how an action is to be used or how actions are connected with each other (Smith 2003, Frank 2003). An important omission was also (considered) the absence of explicit resources, to be defined independently from action pre and post conditions (Geffner 2003). All these features may be essential to represent the inspecton actions efficiently and flexibly.

Inspection HTN

Due to the natural structuring of the inspection duties into the hierarchically, functionally and temporalily related activities we consider as an alternative (and perhaps a better) representation and the planner some version of the Hierarchical Task Networks (HTN) and its planner (e.g. SHOP, SHOP2, JSHOP2 (Nau 2003, Tian 2012).

HTN shows a number of advantages with respect to the non-hierarchical planning (like e.g. PDDL). It is strictly more expressive than classical (non-hierarchical) planning and it allows specifying problems that cannot be expressed using the classical planning formalism. Furthermore the task hierarchy in HTN planning can encode domain-specific search control (Alford 2016). HTN allows plan generation to be limited by external constraints. The decomposition information in HTN plans can also help guide plan execution. HTN planners decompose complex tasks into primitive tasks, building a plan tree, which terminates at leaves that correspond to primitive actions. The usual implementations, e.g. SHOP2 are forward state-exploring planner, keeping at all times a notion of the current state. (Goldman 2006)

There is quite an amount of research on translating the planning problem from HTN to PDDL (Alford 2009) (HTN-PDDL), from PDDL to HTN (Goldman 2006), or combining the hierarchical domain knowledge in the form of Hierarchical Task Networks with domain-independent local search techniques of LPG planner (Gerevini 2008).

The natural hierarchical structure of the HTN problem representation made it possible to experiment with the automated derivation of the HTN graphs from the UML models (Palao 2011) and from flow patterns (Sohrabiy 2012). Special version of the HTN - Hierarchical Agent-based Task Planner (HATP) was also designed which makes the traditional HTN planning domain representation and semantics more suitable for robotics, interleaving planning with geometric reasoning to validate online the robot actions with respect to a detailed 3D world description. HATN permits also to filter plans accordingly to the wasted time, effort balancing, control of intricacy, and violation of specific user-defined sequences. (Lallement 2014)

How could we use the HTN representation to represent inspection duties of the CFR?

Consider for example the HTN-like view of a fragment of the inspection activities at an imaginary equipment inspection point (Fig 14, blue shapes are tasks and sub-tasks, pink shapes are pre-conditions, and green/yellow shapes are primitive operators). The inspection (here the measurement) means the hierarchy and sequence of conditionally dependent smaller measurement tasks. First ambient conditions must be checked (selected ambient variables measured) to decide whether the main measurement can be carried out at all (in strong wind for example measuring the gas leakage may be impossible), and if yes, under what special conditions (in cold weather sensor heating may be needed). Then the calibration of the principal sensor must be checked. If calibration is due, it must be scheduled ahead of the proper measurements and its results must be taken into account (e.g. as correction factors to recompute the inspected quantity). (In the figure vertical arrows indicate further functional branches until the level of the primitive actions, or recursion is reached). HTN representation of the inspection task will be processed by some HTN planner to produce the executable plan of primitive actions. In the figure green color indicates primitive functional actions yet above the ROS message level, and the yellow color indicates plans in terms of direct ROS messages.

This planning problem would be delegated to the Operator Center system and is not discussed in detail here. The preparation of the inspection HTN plans for diverse equipment belonging to the installation is the task of the knowledge engineers on the staff of the Operator Center (or the Remote Operator Center).

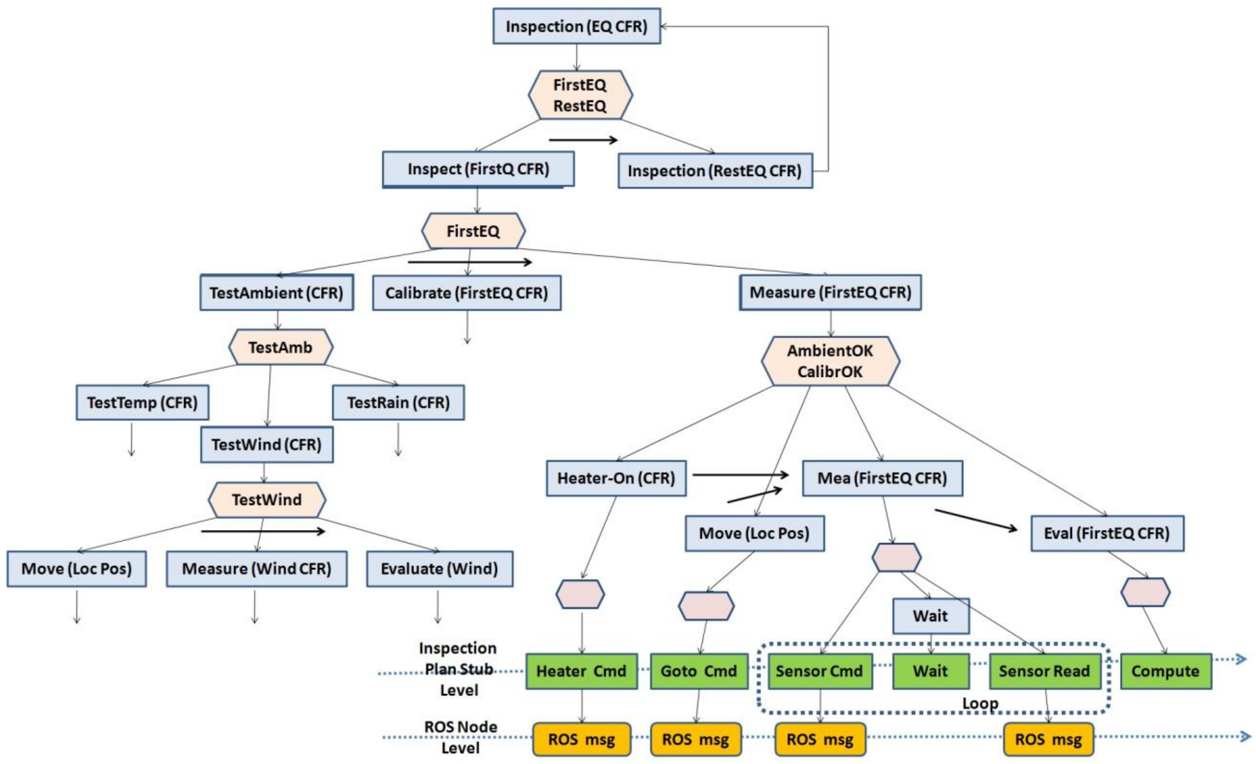


Fig 14. HTN-like view of a fragment of typical measurement-based inspection activities at an imaginary equipment inspection point.

Implementation of a concrete measurement task (here „Measure”) may depend on the implemented sensors, i.e. each kind of sensor would mean a different method for the task (not depicted in the Figure). This additional branching must be included for all the sensor types demanding differentiated management and coming into question in the CFR re-configuration.

Inspection Plan Stubs woud be computed from the HTN-like structure at the OpC system with specialized planner based on installation requirements (what is to be measured at a given equipment inspection point) and based on the CFR configuration (what sensors are installed to realize the measurements, and what are the sensor properties). Stubs contain primitive operations interpretable by the Inspection Planner, serialized, or embedded in the conditional and loop control structures.

Program control structures for the Inspection Plan Stub

Assume that the Inspection Plan Stubs can be expressed as the above HTN structures. A normal HTN planner, like e.g. SHOP (Nau 2003) traces the HTN trees, checking the validity of the preconditions, expanding recursively task definitions and producing a sequence of the primitive actions.

In the present case such planner should be modified to permit in the resulting (primitive action level) plan also (1) conditional and (2) repetitive control structures. Why?

(1) Inspection (measurement) is the context dependent activity and a part of this context is the run-time information (e.g. AmbientOk in the Measure task). In the e.g. gas leakage inspection, sensor must be pre-heated if the ambient temperature is too low. If the particular sensor in the CFR configuration does not require heating, this part of the inspection plan can be skipped. This information is available off-line and can be used to tailor the inspection plan to the CFR configuration prior to the mission. If however the heater is present in the configuration, its maintenance must be included into the inspection plan conditionally, based upon the run-time results of the (also inserted on purpose) additional test measurement of the ambient temperature at the place and the moment of the inspection (on run-time variables see also (Etzioni 1992).

Similar considerations can be done for the windy conditions. If the wind is strong, perhaps the gas sensing must be done from a closer distance, or abandoned at all if the wind is too strong. This however is also run-time information. It may also happen that the CFR should contact the operator on the spot, delegate the decision by conveying the measured ambient conditions, and act upon the received command. The functional dependencies of the metrological situations are full of similar problems. Consider next the similar question of the calibration.

(2) Every measurement is noisy (uncertain, imprecise) to some extent. How to handle the uncertain measurement results is regulated formally by the metrological bodies. For us it is important that the accuracy of the measurement results (and the quality of the decisions based upon them) may be increased on behalf of the measurement time by simple averaging. Averaging means the repetition of the measurement plan, and the number of the averaged data depends upon the noisiness of the environmental situation and the accuracy requirements, both may be run-time dependent.

It is essential thus that the downloaded Inspection Plan (Stub) should resemble a normal program control flow, with the conditionals and loop structures containing also run-time variables.

These problems require basic research at the border of the AI Planning, robotics, and the measurement theory. Before any essential break-through will be done and before such planning system becomes a mass tool in robotics, it is safe to assume that that kind of plans will be computed and adjusted manually, within the expertise ring of the immediate system designers.

Re-use of the Inspection HTN

An interesting problem is the expected similarity of the inspection plans for technically similar equipment or equipment differing only in the parameter setting. Inspection plans for such equipment can be reused by being tailored and parameterized. This can be done manually, but perhaps a more serious solution may be to use the recently developed Description Logic (DL) based HTN planner (Hartando 2008, 2009). In this approach the particular planning problem (equipment case) is described formally in (description) logic. The reasoning with the logical knowledge base (so called T-Box and A-Box) is used to reduce the full HTN planning problem to the needs of the particular equipment, which is then solved by a normal HTN planner transforming the generic HTN inspection plan to the particular inspection case. A further interesting development would be to fuse this approach with the HATN planning framework (Lallement 2014), better oriented to the needs of the robotic systems.

ROS-level actions or not?

If the inspection plans would be maintained in the HTN-like representation, one of the principal design questions would be the choice of the primitive actions (operators) – leaves where the HTN branches end.

There are two possible choices:

(1) Generic set of inspection related primitive actions in the leaves, like e.g. ’set parameter’, ’start ADC conversion’, ’switch on/off’, ’read out value’, ’wait transient’, ’apply excitation’, etc. drawing from the sensor ontology and the general metrology. Their advantage would be the lasting general applicability of the HTN representation, not only to the inspection robots, but to planning any kind of the physical experiments, but such primitive actions wouldn’t be directly applicable as the commands to the measurement hardware (Fig 14, green level).

To execute the inspection plans the primitive actions should be translated in the Inspection Planner into low-level control commands, in our case into the ROS message sequences. Such translation would assume in the KDB existence of a primitive action-to-ROS dictionary which should be occasionally updated to the actual assortment and version of the used ROS packages.

(2) The other solution is to develop the HTN representation all the way down to the ROS message level. In this case the plan developed from the HTN representation by the HTN planner would be immediately executable. The disadvantage of such approach is narrowing the scope of the represented inspection plans (to the robot configurations) and every now and then checking the consistency of the representation with the changes at the ROS package level (Fig 14, yellow level).

5.3 Learning the mission plan?

An interested option to plan the mission and the inspection (advocated for the offshore inspection in the literature (Graf 2007)) is to lead the robot along the installation by teleoperating, issuing movement, sensing and inspection commands, storing them and processing later into a plan which is filled then to the robot for an autonomous execution (The paper does not dwell upon what such a storage could be, see also (Argall 2009) for an interesting review, yet without a clear solution for the CFR problem).

As far as such plan would be easy to record, it would be exceptionally „brittle” in a sense that it would be very difficult to modify (re-planning, re-configuration) to the changing conditions in the sensory apparatus (e.g. a new technology of sensors), in the mission (e.g. different route, permanently obstacle passages due to the prolonged maintenance), or in the environment (e.g. plan recorded in good weather and to be executed in bad weather).

Any such re-use of the plan would call for a serious plan pre-processing, annotating the plan steps with goal comments, indicating dependencies, and abstracting plan sections into higher level commands. All this would require a knowledge intensive workstation and an expertise on behalf of the operator. Frankly speaking the AI literature is poor in bottom-up synthesis and planning methods. For the hierarchical planning the other way round is cultivated.

The normal approach to learning through repetition of examples is out of question in the CFR domain. Yet another problem is the learning of conditional plan branches for rare, but mission critical events (related to the alarms, misfunctioning, or extreme weather conditions) (Sermanet 2008).

In addition in the onshore installations the lay-out is more clear-cut, can be better modeled and the plans can be computed based on the path graphs and the functional requirements.

6 References

(Akin Sisbot 2007) E. Akin Sisbot, L.F. Marin-Urias, R. Alami, and T. Simeon, **A Human Aware Mobile Robot Motion Planner**, IEEE Trans. on Robotics, Vol 23 N°5 (2007)

(Alford 2009) Alford, R.; Kuter, U.; and Nau, D. S., **Translating HTNs to PDDL: A small amount of domain knowledge can go a long way**, Proc. of IJCAI, 1629–1634, 2009.

(Alford 2016) R. Alford, G. Behnke, D. Höller, P. Bercher, S. Biundo, D.W. Aha, **Bound to Plan: Exploiting Classical Heuristics via Automatic Translations of Tail-Recursive HTN Problems**, ICAPS 2016, the 26th Int. Conf. on Automated Planning and Scheduling, London, UK, June 12–17, 2016.

(Arain 2015a) Arain, M A., Cirillo, M., Hernandez Bennetts, V., Schaffernicht, E., Trincavelli, M. et al., **Efficient Measurement Planning for Remote Gas Sensing with Mobile Robots,** 2015 IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 3428-3434, Washington, 2015

(Arain 2015b) M.A. Arain, M. Trincavelli, M. Cirillo, E. Schaffernicht, and A.J. Lilienthal, **Global Coverage Measurement Planning Strategies for Mobile Robots Equipped with a Remote Gas Sensor**, Sensors (Basel). 2015 Mar; 15(3): 6845–6871.

(Argall 2009) B.D. Argall, S. Chernova, M. Veloso, B. Browning, **A survey of robot learning from demonstration**, Robotics and Autonomous Systems 57 (2009) 469483.

(Bandyopadhyay 2013) T. Bandyopadhyay, C. Zhuang Jie, D. Hsu, M.H. Ang Jr., D. Rus, and E. Frazzoli, **Intention-Aware Pedestrian Avoidance**, Experimental Robotics, Vol 88, Springer Tracts in Advanced Robotics, pp. 963-977, 2013

(Barber 2015) R. Barber, M. A. Rodriguez-Conejo, J. Melendez and S.o Garrido, **Design of an Infrared Imaging System for Robotic Inspection of Gas Leaks in Industrial Environments**, Int. J. of Advanced Robotic Systems, Vol 12, Number 23, 2015.

(Bennetts 2015) Bennetts, **Mobile Robots with In-situ and Remote sensors**, Örebro, PhD, 2015.

(Bogomolov 2003) Y. Bogomolov, G. Dror, S. Lapchev, E. Rivlin, M. Rudzsky, **Classification of Moving Targets Based on Motion and Appearance**, British Machine Vision Conf. 2003, pp. 1-10, 2003.

(Buisson 2013) J. Buisson, S. Galland , N. Gaud , M. Goncalves , A. Koukam, **Real-time Collision Avoidance for Pedestrian and Bicyclist Simulation: a smooth and predictive approach**, Procedia Computer Science 19 (2013) 815–820 (The 2nd Int. Workshop on Agent-based Mobility, Traffic and Transportation - Models, Methodologies and Applications)

(Cacace 2014) J. Cacace, A. Finzi, V. Lippiello, G. Loianno, D. Sanzone, **Aerial service vehicles for industrial inspection: task decomposition and plan execution**, Appl Intell (2015) 42:49–62

(Cambon 2004a) S. Cambon, F. Gravot, R. Alami, **A robot task planner that merges symbolic and geometric reasoning**, 2004/8/22, ECAI, Vol 16, pp. 895.

(Cambon 2004b) S. Cambon, F. Gravot, R. Alami, **aSyMov: Toward More Realistic Robot Plans**, Int. Conf. on Automated Planning and Scheduling, 2004.

(Cambon 2009)S. Cambon, R. Alami, F. Gravot, **A hybrid approach to intricate motion, manipulation and task planning**, The Int. J. of Robotics Research, Vol 28, Nr 1, pp. 104-126, 2009.

(Cashmore 2015) M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, **ROSPlan: Planning in the Robot Operating System**, Proc. of the Twenty-Fifth Int. Conf. on Automated Planning and Scheduling, ICAPS 2015.

(Chen 2014a) H. Chen, S. Stavinoha, M. Walker, B. Zhang, T. Fuhlbrigge, **Opportunities and Challenges of Robotics and Automation in Offshore Oil & Gas Industry**, Intelligent Control and Automation, **5**, 136-145, 2014.

(Chen 2014b) H. Chen, S. Stavinoha and Michael Walker, Biao Zhang and Thomas Fuhlbrigge, **Exploring Robotic Applications in Offshore Oil & Gas Industry**, 2014 IEEE 4th Ann. Int. Conf. on Cyber Techn. in Automation, Control, and Intelligent Systems (CY-BER), 4-7 June 2014, pp. 563-568, Hong Kong.

(Cho 2010) H. Cho, P.E. Rybski, W. Zhang, **Vision-based Bicyclist Detection and Tracking for Intelligent Vehicles**, Intelligent Vehicles Symposium (IV), 2010 IEEE, 454-461.

(Codd-Downey 2014) R. Codd-Downey, M. Jenkin and A. Speers, **Building a ROS Node for a NMEA Depth and Temperature Sensor**, 2014 11th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO), 1-3 Sept 2014, 506–512, Vienna.

(de Carvalho 2013) G.P.S. de Carvalho, G.M. Freitas, R.R. Costa, G.H.F. de Carvalho, J.F.L. de Oliveira, S.L. Netto, E.A.B. da Silva, M.F.S. Xaud, L. Hsu, G. Motta-Ribeiro, A.F. Neves, F.C. Lizarralde, I. Marcovistz, A.J. Peixoto, E.V.L. Nunes, P.J. From, M. Galassi, and A. Roy-roy, **DORIS - Monitoring Robot for Offshore Facilities**, Offshore Techn. Conf., Rio de Janeiro, Brazil, 29–31 Oct, 2013, OTC OTC-OTC-24386-MS.

(de Silva 2013) L. de Silva, A. Kumar Pandey, M. Gharbi, R. Alami, **Towards Combining HTN Planning and Geometric Task Planning**, RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications, June 2013

(Distante 2009) Distante C., G. Indiveri and G. Reina, **An application of mobile robotics for olfactory monitoring of hazardous industrial sites**, Int. J. Industrial Robot, 36/1, 2009, 51–59

(Edelkamp 2015) S. Edelkamp, E. Plaku , C. Greulich, M. Pomarlan, **Solving the Inspection Problem via Colored Traveling Salesman Tours,** IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) - Workshop on Task Planning for Intelligent Robots in Service and Manufacturing, Hamburg, 2015.

(Englot 2011) B. Englot and F. Hover, **Planning Complex Inspection Tasks Using Redundant Roadmaps**, Proc. Int. Symp. Robotics Research, 2011

(Englot 2012) B. Englot and F. Hover, **Sampling-Based Coverage Path Planning for Inspection of Complex Structures**, Proc. of the Twenty-Second Int. Conf. on Automated Planning and Scheduling, 2012

(Etzioni 1992) O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, M. Williamson, **An Approach to Planning with Incomplete Information,** Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning, 1992

(Faigl 2011) J. Faigl, M. Kulich, L. Preucil, **A Sensor Placement Algorithm for a Mobile Robot Inspection Planning**, J. Intell. Robot Syst. (2011) 62:329–353

(Fox 1997) Fox, D., Burgard W:, Thrun S., **The dynamic window approach to collision avoidance**, IEEE Robotics & Automation Mag., Vol 4, Issue 1, Mar 1997, pp. 23 - 33.

(Fox 2003) Fox, M., and Long, D., **PDDL2.1: An extension to PDDL for expressing temporal planning domains,** J. of Artificial Intelligence Research 20:61–124, 2003.

(Frank 2003) J. Frank, K. Golden, A. Jonsson, **The Loyal Opposition Comments on Plan Domain Description Languages**, Int. Conf. on Artificial Intelligence Planning and Scheduling, June 2003

(Galceran 2013) E. Galceran, M. Carreras, **A survey on coverage path planning for robotics**, Robotics and Autonomous Systems 61 (2013) 1258–1276

(Geffner 2003) H. Geffner, **Commentary, PDDL 2.1: Representation vs. Computation**, J. of Artifitial Intelligence Research 20 (2003) 139-144

(Geraerts 2007) R. Geraerts and M.H. Overmars, **The corridor map method: a general framework for real-time high-quality path planning**, Comp. Anim. Virtual Worlds 2007; 18: 107–119

(Geraerts 2010) R. Geraerts, **Planning Short Paths with Clearance using Explicit Corridors**, IEEE Int. Conf. on Robotics and Automation, pp. 1997-2004, 2010.

(Gerevini 2008) A. Gerevini, U. Kuter, D. Nau, A. Saetti, N. Waisbrot, **Combining Domain-Independent Planning and HTN Planning: The Duet Planner**, Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), Sept. 2008.

(Goldman 2006) R.P. Goldman, **Durative Planning in HTNs**, Proc. of ICAPS 2006.

(Graf 2007) B. Graf, K. Pfeiffer, H. Staab, **Mobile Robots for Offshore Inspection and Manipulation**, Int. Petroleum Technology Conf., Dubai, U.A.E., Dec 4-6, 2007

(Gutin 2006) Gutin G., Punnen A.P., **The Traveling Salesman Problem and Its Variations**, Springer, 2006

(Guzzi 2013) J. Guzzi, A. Giusti, L.M. Gambardella, G. Theraulaz, G. A. Di Caro, **Human-friendly Robot Navigation in Dynamic Environments**, 2013 IEEE Int. Conf. on Robotics and Automation (ICRA), 6-10 May 2013, pp. 423–430, Karlsruhe.

(Habib 2014) Habib A., G. Bonov, A. Kroll, J. Hegenberg, L. Schmidt, T. Barz, D. Schulz, **ROBOGAS$^{\text{INSPECTOR}}$ Research Project: Detecting Gas Leaks with Autonomous Mobile Robots**, Ex-MAGAZINE 2014, pp. 91-97

(Hamasaki 2011) S. Hamasaki, Y. Tamura, A. Yamashita and H. Asama, **Prediction of Human's Movement for Collision Avoidance of Mobile Robot**, 2011 IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), 7-11 Dec 2011, pp. 1633–1638, Karon Beach, Phuket

(Hansen 2007) E.A. Hansen, R. Zhou, **Anytime Heuristic Search**, J. of AI Res., 28 (2007) 267-297

(Hartando 2009) R. Hartando, **Fusing DL Reasoning with HTN Planning as a Deliberative Layer in Mobile Robotics**, PhD, Univ. Osnabrück.

(Hartando 2009) R. Hartando, J. Hertzberg, **Fusing DL Reasoning with HTN Planning**, in KI 2008: Advances in Artificial Intelligence, Vol 5243 of the series Lecture Notes in Computer Science, pp. 62-69

(Helsgaun 2006) K. Helsgaun, **An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic**, Writings on Computer Science, No. 109, Roskilde Univ., 2006

(Helsgaun 2014) K. Helsgaun, **Solving the Clustered Traveling Salesman Problem Using the Lin-Kernighan-Helsgaun Algorithm**, May 2014, Roskilde Univ. Comp. Sc. Res. Rep. #142

(HTN-PDDL) **Translating HTN planning problems to PDDL**, https://github.com/ronwalf/HTN-Translation

(Hussain Ahmed 2014) Z. Hussain Ahmed, **The Ordered Clustered Travelling Salesman Problem: A Hybrid Genetic Algorithm**, Scientific World J., Vol 2014, Article ID 258207

(Ibarguren 2013) A. Ibarguren, J. Molina, L. Susperregi and I. Maurtua, **Thermal Tracking in Mobile Robots for Leak Inspection Activities**, Sensors 2013, 13, 13560-13574

(Jordan 1993) M. I. Jordan, R. A. Jacobs, **Hierarchical mixtures of experts and the EM algorithm**, Proc. of 1993 Int. Joint Conf. on Neural Networks, 25-29 Oct, Nagoya, 1993.

(Kim 2011) J.-H. Kim, **Sensor-based Autonomous Pipeline Monitoring Robotic System**, PhD, 2011, Louisiana State Univ. and Agricultural and Mechanical College

(KML) **Keyhole Markup Language**, ttps://en.wikipedia.org/wiki/Keyhole_Markup_Language

(Kortenkamp 2013) D. Kortenkamp, R. Simmons, and D. Brugali, **Robotic systems architectures and programming**, Springer Handbook of Robotics – 2nd Ed, 2013.

(Kroll 2008) Kroll A., **A survey on mobile robots for industrial inspection**, Int. Conf. on Intelli. Autonomous Syst. (IAS10), pp. 406-414, 23-25 July, 2008.

(Kyrkjebo 2009) E. Kyrkjebo, P. Liljeback, and A. A. Transeth, **A robotic concept for remote inspection and maintenance on oil platforms**, Proc. ASME 28th Int. Conf. on Ocean, Offshore and Arctic Engineering (OMAE 2009), Hawaii, USA, May 31 - June 5, 2009.

(Lallement 2014) R. Lallement, L. de Silva, R. Alami, **HATP: An HTN Planner for Robotics**, Proc. PlanRob 2014.

(Leibold 2012) S. Leibold, A. Fregin, D. Kaczor, M. Kollmitz, K. El Menuawy, E. Popp, J. Kotlarski, J. Gaa, B. Munske, **RoboCup@ Work League Winners 2012,** RoboCup 2012: Robot Soccer World Cup XVI, Vol 7500 LNCS Series, pp. 65-76, 2012.

(Leon-Rodriguez 2013) H. Leon-Rodriguez, T. Sattar, J.-O. Park, **Wireless Climbing Robots for Industrial Inspection**, 2013 44th Int. Symp. on Robotics (ISR), 24-26 Oct, 2013, pp. 1-4, Seoul.

(Li 2015) J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, **Colored Traveling Salesman Problem**, IEEE Trans. on Cybernetics, Vol. 45, No. 11, Nov 2015.

(Likhachev 2010) M. Likhachev, **Search-based Planning with Motion Primitives**, Tutorial at CoTeSys-ROS Fall School on Cognition-enabled Mobile Manipulation, Munich, 2010, http://www.cs.cmu.edu/~maxim/files/tutorials/robschooltutorial_oct10.pdf

(Liu 2012) X. Liu, S. Cheng, H. Liu, S. Hu, D. Zhang and H. Ning, **A Survey on Gas Sensing Technology**, Sensors 2012, 12, 9635-9665

(Lu 2014) D. V. Lu, D. Hershberger, W. D. Smart, **Layered Costmaps for Context-Sensitive Navigation,** 2014 IEEE/RSJ Int. Conf. on Intell. Robots and Syst., 14-18 Sept, 2014, pp. 709-715, Chicago, IL.

(Martínez-Gómez 2014) J. Martínez-Gómez, A. Fernández-Caballero, I. García-Varea, L. Rodríguez and C. Romero-González, **A Taxonomy of Vision Systems for Ground Mobile Robots**, Int J Adv Robot Syst, 2014, 11:111.

(Mataric 2002) [17] M. Mataric et al., **Situated robotics**, Encyclopedia of Cognitive Science, vol. 4, pp. 25-30, 2002.

(Maurtuaa 2014) Maurtuaa I., L. Susperregia, A. Fernándeza, C. Tubíoa, C. Perezb, J. Rodríguezc, T., Felschd, M. Ghrissie, **MAINBOT – mobile robots for inspection and maintenance in extensive industrial plants**, Energy Procedia 49 (2014) 1810–1819, SolarPACES 2013

(Morignot 2010) P. Morignot, M. Soury, C. Leroux, H. Vorobieva, P. Hède, **Generating scenarios for a mobile robot with an arm**. Case study: Assistance for handicapped persons, 11th Int. Conf. on Control, Automation, Robotics and Vision (ICARCV'10), Singapore, Dec 2010, P1054

(Morignot 2011) P. Morignot, M. Soury, P. Hède, C. Leroux, **Generating and executing scenarios for a mobile robot**, Proc. of the Jochen Pfalzgraf Memorial Symposium (5th Symposium on Multiagent Systems, Robotics and Cybernetics, InterSymp-2011), 23rd Int. Conf. on Systems Research, Informatics and Cybernetics (IIAS'11), G. E. Lasker and O. Bartheye eds., Baden-Baden, Germany, August 3, 2011

(Moussaid 2011) M. Moussaid, D. Helbing, G. Theraulaz and S. Hanson, **How simple rules determine pedestrian behavior and crowd disasters**, Proc. of the Nat. Acad. of Sciences of the United States of America, Vol. 108, No. 17 (April 26, 2011), pp. 6884-6888

(Nau 2003) D. Nau, Tsz-Chiu, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, F. Yaman, **SHOP2: An HTN Planning System**, J. of Artificial Intell. Research 20 (2003) 379-404

(Pacchierotti 2005) E. Pacchierotti, H.I. Christensen, and P. Jensfelt, **Embodied social interaction for service robots in hallway environments**, Proc. of the 5th Int. Conf. on Field and Service Robotics (FSR), 2005.

(Palao 2011) F. Palao, J. Fdez-Olivares, L. Castillo, O. Garcia, **An extended HTN knowledge representation based on a graphical notation**, KEPS 2011: ICAPS Workshop on Knowledge Engineering for Planning and Scheduling

(Pan 2012) J. Pan, S. Chitta, D. Manocha, **FCL: A General Purpose Library for Collision and Proximity Queries,** 2012 IEEE Int. Conf. on Robotics and Automation (ICRA), 14-18 May 2012, pp. 3859-3866, Saint Paul, MN.

(PDDL) **Planning Domain Definition Language**, https://en.wikipedia.org/wiki/Planning_Domain_Definition_Language

(Petereit 2012) J. Petereit, T. Emter, C. W. Frey, T. Kopfstedt, A. Beutel, **Application of Hybrid A* to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor**

Environments, Proc. of ROBOTIK 2012, 7th German Conf. on Robotics, 21-22 May 2012, pp. 1-6.

(Pierce 2014) S.G. Pierce, C.N. Macleod, G. Dobie, R. Summan, **Path Planning & Measurement Registration for Robotic Structural Asset Monitoring**, 7th Europ. Workshop on Structural Health Monitoring, July 8-11, 2014. La Cité, Nantes, France

(PostgreSQL) **PostgreSQL 9.5.0 Documentation**, http://www.postgresql.org/docs/

(R5-COP D11.10) R5-COP D11.10 **Scenarios and use cases for demonstrators**

(R5-COP D26.11) R5-COP_D26.11 **Planning and Reasoning Architecture**

(R5-COP D26.30) D26.30 **Task planning and representation**

(R5-COP D41.10) R5-COP D41.10 **Application scenarios and requirements, Industrial robots demonstration 3: Robot co-worker**

(R5-COP D41.20) R5-COP D41.20 **Integration of configurable components and application development**

(Reggente 2009) M. Reggente and A.J. Lilienthal, **Using Local Wind Information for Gas Distribution Mapping in Outdoor Environments with a Mobile Robot**, IEEE Sensors 2009, 25-28 Oct, 2009, pp. 1715-1720, Christchurch

(Reyes Ballesteros 2012) A. Reyes Ballesteros, A. Felix, C. Gonzales and E. Islas Perez, **Optimal robot navigation for inspection and surveillance in electronic substations**, CIGRE Int. Symp., Paris, August 2012

(Robogasinspector, 2013) http://www.robogasinspector.de/

(Robotnik) **Mobile robot SUMMIT XL HL**, http://www.robotnik.eu/mobile-robots/summit-xl-hl/

(ROS actionlib) http://wiki.ros.org/actionlib

(ROS Adding Relaxed Astar) http://www.iroboapp.org/index.php?title=Adding_Relaxed_Astar_Global_Path_Planner_As_Plugin_in_ROS

(ROS arduino Measuring Temperature) http://wiki.ros.org/rosserial_arduino/Tutorials/Measuring%20Temperature

(ROS base_local_planner) http://wiki.ros.org/base_local_planner

(ROS carrot_planner) http://wiki.ros.org/carrot_planner

(ROS cmd_vel_mux) http://wiki.ros.org/cmd_vel_mux

(ROS costmap_2d) http://wiki.ros.org/costmap_2d

(ROS costmap_2d/hydro/inflation) http://wiki.ros.org/costmap_2d/hydro/inflation

(ROS costmap_2d/hydro/obstacles) http://wiki.ros.org/costmap_2d/hydro/obstacles

(ROS costmap_2d/hydro/staticmap) http://wiki.ros.org/costmap_2d/hydro/staticmap

(ROS dwa_local_planner) http://wiki.ros.org/dwa_local_planner

(ROS eband_local_planner) http://wiki.ros.org/eband_local_planner

(ROS Global Planner As Plugin) http://wiki.ros.org/navigation/Tutorials/Writing%20A%20Global%20Path%20Planner%20As%20Plugin%20in%20ROS

(ROS global_planner) http://wiki.ros.org/global_planner

(ROS map_server) http://wiki.ros.org/map_server

(ROS nav_core) http://wiki.ros.org/nav_core

(ROS navfn) http://wiki.ros.org/navfn

(ROS Navigation Stack) http://wiki.ros.org/navigation/Tutorials/RobotSetup

(ROS nxt_ros) http://wiki.ros.org/nxt_ros

(ROS phidgets_ros) http://wiki.ros.org/phidgets_ros

(ROS pluginlib) http://wiki.ros.org/pluginlib

(ROS robot_localization) http://wiki.ros.org/robot_localization

(ROS rosserial Humidity)
http://wiki.ros.org/rosserial_mbed/Tutorials/DHT%20Humidity%20and%20Temperature%20Sensor

(ROS rosserial Measuring Temperature)
http://wiki.ros.org/rosserial_mbed/Tutorials/Measuring%20Temperature

(ROS sbpl) http://wiki.ros.org/wiki/sbpl

(ROS Sensors) http://wiki.ros.org/Sensors

(ROS Services) http://wiki.ros.org/Services

(ROS teb_local_planner) http://wiki.ros.org/teb_local_planner

(ROS Topics) http://wiki.ros.org/Topics

(ROS yocs_cmd_vel_mux) http://wiki.ros.org/yocs_cmd_vel_mux

(ROS yocs_velocity_smoother) http://wiki.ros.org/yocs_velocity_smoother

(Russell 2005) Russell S., Norvig P., **Artificial Intelligence: A Modern Approach**, 2nd Ed., Prentice Hall, 2005

(SAM) **SAM, A service robot for assisting disabled/ageing persons**, July 12, 2010, http://philippe.morignot.free.fr/SAM/

(Sensabot 2012) **Sensabot: A Safe and Cost-Effective Inspection Solution**, J. of Petroleum Technology, Vol 64, Issue 10, October 2012.

(Sermanet 2008) P. Sermanet, M. Scoffier, C. Crudele, U. Muller, Y. LeCun, **Learning Maneuver Dictionaries for Ground Robot Planning**, Proc. 39th Int. Symp. on Robotics (ISR'08), 2008

(Shi 2008) D. Shi, E.G. Collins Jr., A. Donate, X. Liu, B. Goldiez, D. Dunlap, **Human-Aware Robot Motion Planning with Velocity Constraints**, CTS 2008, Int. Symp. on Collaborative Technologies and Systems, 19-23 May 2008, pp. 490–497, Irvine, CA.

(Shukla 2016a) A. Shukla, H. Karki, **Application of robotics in onshore oil and gas industry- A review Part I**, J. Robotics and Autonomous Syst., Vol 75, Issue PB, Jan 2016, pp. 490-507

(Shukla 2016b) A. Shukla, H. Karki**, Application of robotics in offshore oil and gas industry- A review Part II**, J. Robotics and Autonomous Syst., Vol 75, Issue PB, Jan 2016, pp. 508-524

(Smith 2003) D.E. Smith, **The Case for Durative Actions: A Commentary on PDDL2.1**, Journal of Artificial Intelligence Research 20, pp. 149-154, 2003.

(Sohrabiy 2012) S. Sohrabiy, O. Udrea, A. Ranganathan, A.V. Riabov, **Composition of Flow-Based Applications with HTN Planning Problem Solving Using Classical Planners**, AAAI Technical Report WS-12-12, 2012.

(Soldan 2012) Soldan, S.; Bonow, G. and Kroll, A., **RoboGasInspector – A Mobile Robotic System for Remote Leak Sensing and Localization in Large Industrial Environments: Overview and First Results**, Proc. of the 2012 IFAC Workshop on Automatic Control in Offshore Oil and Gas Production, pp. 33-38, 31.05 - 01.06., 2012.

(Spiewak 2015) A. Spiewak, W. Salabun, **A Mobile Gas Detector with an Arduino Microcontroller**, Int. J. Computer Technology & App., Vol 6 (4), 636-641, 2015

(Steele 2014) J.P. H. Steele, Q. Han, H. Karki, K. Al-Wahedi, A.A. Ayoade, M.R. Sweatt, D.P. Albert, W. A. Yearsley, **Development of an Oil and Gas Refinery Inspection Robot**, ASME 2014 Int. Mech. Eng. Cong. and Exposition, Vol 4A: Dynamics, Vibration, and Control, Montreal, Quebec, Nov 14–20, 2014, Paper No. IMECE2014-36358.

(Surmann 2008) H. Surmann, D. Holz, S. Blumental, T. Linder, P. Molitor, V. Tretyakov, **Teleoperated Visual Inspection and Surveillance with Unmanned Ground and Aerial Vehicles**, Int. J. of Online Eng. (iJOE), Vol 4, No 4 (2008).

(TALON 2004) **TALON Robot Detects Chemicals, Gases, Radiation and Heat**, Waltham MA (SPX) Sep 28, 2004

(Tamura 2010) Y. Tamura, T. Fukuzawa, H. Asama, **Smooth collision avoidance in human-robot coexisting environment**, The 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Oct 18-22, 2010, Taipei, Taiwan

(Thayer 2010) J. Thayer and W. Ruml, **Anytime Heuristic Search: Frameworks and Algorithms**, Proc. of the Third Ann. Symp. on Combinatorial Search (SOCS-10), 2010.

(Tian 2012) J. Tian, Z. Li, **Emergency Tasks Planning Based on Formal Modeling of Emergency Plan and HTN Planning System SHOP2**, Intelligent Information Management, 2012, 4, 357-363

(Tomović 2012) A. Tomović, **Path Planning Algorithms For The Robot Operating System,** 2014 Midwest Instruction and Computing Symp. (MICS 2014), Verona, WI

(Ulrich 2016) M. Ulrich, G. Lux, L. Jurgensen, G. Reinhart, **Automated and Cycle Time Optimized Path Planning for Robot-Based Inspection Systems**, Procedia CIRP 44 (2016) 377 – 382 (6th CIRP Conf. on Assembly Technologies and Systems (CATS))

(Vasquez 2013) D. Vasquez, P. Stein, J. Rios-Martinez, A. Escobedo, A. Spalanzani and C. Laugier, **Human Aware Navigation for Assistive Robotics,** in Experimental Robotics, Vol 88, Springer Tracts in Advanced Robotics, pp. 449-462, 2013.

7 Appendix A

As a feasibility study for the multi-layered local costmap proposed in the Section 4.2.4 as a component of the extended Navigation Planner we analyze the hallway traffic situation when the CFR catches up with a slow pedestrian, has intention to overtake, but a bicyclist is drawing near from ahead, see Fig 15.

We assume that the sensory apparatus and the Sensor Fusion Module of the CFR can compute the respective $x_P, x_C, v_P, v_C, \quad v_P < v_R$ positions and velocities of the pedestrian (dynamic obstacle in front) and the cyclist (potential dynamic obstacle from ahead). The v_R velocity of the CFR is also known and its position is assumed to be 0. We also know d_P, d_C the personal spaces of the pedestrian and the cyclist (the last one is larger due to the speedy movement). Let *D* denote the width of the road and d_R the space demand of the robot (its personal space).

When we start this analysis, we assume that: $v_P < v_R, \quad x_P - d_P \geq x_R$ (A.1)

If the cyclist is far enough, the CFR will switch to the left lane; overtake the pedestrian using the speed difference, and cut in ahead of the pedestrian, still observing a satisfactory free margin to the cyclist. Let the time-point of cutting-in be t' and the positions by then of all the participants be: x'_R, x'_P, x'_C. Then the situation can be described as follows:

$$\begin{aligned} x'_P &= x_P + t' v_P + d_P \\ x'_C &= x_C - t' v_C - d_C \\ x'_C &\geq x'_P + d_R \end{aligned} \tag{A.2}$$

Let $t_1, t_2, t_3, \quad t' = t_1 + t_2 + t_3$ be the times of switching to the left lane; overtaking the pedestrian, and cutting back in, then:

$$\begin{aligned} t_1\, v_R &= t_3\, v_R \sqrt{2}\, D/2 \\ t_2\, v_R &= x'_P - D/2 \end{aligned} \tag{A.3}$$

Let $D' = \dfrac{2\sqrt{2}-1}{2} D$, with it: $t' = \dfrac{D'}{v_R} + \dfrac{x'_P}{v_R}$. (A.4)

After substitution the condition (A.2) yields the following "overtaking test" computed from a priori known and observable data:

$$\begin{aligned} &(x_C - \alpha x_P) - D' \frac{(v_C + \alpha\, v_P)}{v_R} - (d_C + d_R + \alpha\, d_P) \geq 0 \\ &\alpha = \frac{v_C + v_R}{v_R - v_P} \end{aligned} \tag{A.5}$$

It is interesting to note that the condition is the more difficult to fulfill, the higher the value of α. α will be high, if the robot and the cyclist rush against each other, or when the robot is only sparingly faster than the pedestrian, making the overtaking lengthy.

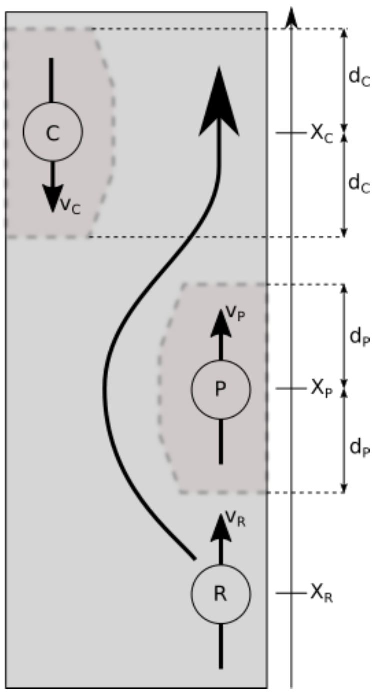


Fig 15. Potential traffic congestion when overtaking slow pedestrians. Darker areas mean personal spaces.

Consider a concrete numerical example.

Let $x_P = 3\frac{m}{\sec}, v_R = 1\frac{m}{\sec}, v_P = 0.25\frac{m}{\sec}, d_C = 1{,}5m, d_P = 1m, d_R = 0{,}5m, D = 3m$.

In this case $\alpha = \frac{4}{3}(1 + v_C)$, and the condition (A.5) simplifies to: $x_C - 8v_C - 12 \geq 0$.

If the cyclist is stationary, the robot can use mere 12 m distance to overtake, but if the cyclist is pressing, 30 m distance will be safe.