A Collaborative Framework for 3D Mapping using Unmanned Aerial Vehicles

P. Doherty[1], J. Kvarnström[1],P. Rudol[1], M. Wzorek[1], G. Conte[1], C. Berger[1], T.Hinzmann[2] and T. Stastny[2]

[1] Dept. of Computer and Information Science, Linköping University, Sweden
[2] Autonomous Systems Lab, ETH Zurich, Switzerland

Abstract. This paper describes an overview of a generic framework for collaboration among humans and multiple heterogeneous robotic systems based on the use of a formal characterization of delegation as a speech act. The system used contains a complex set of integrated software modules that include delegation managers for each platform, a task specification language for characterizing distributed tasks, a task planner, a multi-agent scan trajectory generation and region partitioning module, and a system infrastructure used to distributively instantiate any number of robotic systems and user interfaces in a collaborative team. The application focusses on 3D reconstruction in alpine environments intended to be used by alpine rescue teams. Two complex UAV systems used in the experiments are described. A fully autonomous collaborative mission executed in the Italian Alps using the framework is also described.

Keywords: Emergency and Disaster Management, Robotics and Multi-Robot Systems, Human-Robot Interaction, Distributed Planning and Collaboration, Teamwork, Unmanned Aircraft Systems, Middleware and Interaction Protocols

1 Introduction

Emergency Informatics is an emerging multi-disciplinary scientific field that "*addresses the information processes (real-time collection, analysis, distribution and visualization) for the prevention, preparedness, response and recovery from emergencies*" [1]. In this context, emergencies range from local events to large scale catastrophes. An important and essential aspect of emergency informatics is collaboration and cooperation, where people interact with technical systems either directly or remotely, people interact with people through technical systems, and people interact with remote environments through technical systems such as ground or aerial robots.

This discipline targets highly complex problems characterized by multiple information interdependencies, temporal and spatial scales and latencies, nonlinear behavior and the rarity of optimal solutions to the many problems involved. Consequently, an integrated systems of systems approach to the development of technical systems interacting with emergency responders with the goal of saving lives becomes paramount.

The purpose of this paper is to present an overview of an integrated system of systems of heterogeneous autonomous unmanned aerial vehicles that collaborate with each other and with human responders at a high-level of autonomy with the specific application

goal of generating 3D models of operational environments. These models serve as the basis for the initial level of a dynamic information system through which additional semantic information about the operational environment can be dynamically added and used by first responders. The framework is more general, but the focus in this paper will be on a particular application.

The major components in the system that will be described in this paper are:

- A **Delegation Framework** – This is a speech act based system allowing agents and humans to collaborate by delegating tasks to each other, one-to-one and recursively.
- **Task Specification Trees** – This is the basic construct for representing complex tasks, both declaratively for formal analysis and procedurally as part of the implementation of the delegation framework.
- A **Planning Module** – This module contains a task planner and includes specialized modules associated with the scanning applications that include a scan trajectory generation module and a region partitioning module for distributed scanning.

Each component will be described at an adequate level of detail, both formally and pragmatically, and additional focus will be placed on the integration of these components into a system of systems showing how human responders interact with this technology. Although this paper is applied in nature, emphasis will also be placed on the formal foundations for each of the components. The system itself has been deployed using real UAV (unmanned aerial vehicle) platforms and can be used for relatively complex scanning missions, not only for 3D surface reconstruction, but also for search and rescue. For this paper, we will focus on using a rotor-based and a fixed-wing UAV, but the system is general enough to be used on most any robotic system. The prototype software instantiation of these components is currently being developed for use in the EU project SHERPA ([2], www.sherpa-project.eu). which focuses on the use of heterogeneous robot teams that assist alpine rescuers in the Italian alps in search and rescue missions.

Paper outline. Section 2 begins by describing the basic collaborative mission cycle between humans and robotic systems. Section 3 provides an overview of the complex collaborative 3D reconstruction mission that will be the focus of this paper. Additionally, the two UAV platforms used in the mission will be described. Section 4 describes the delegation framework itself. This section also includes a description of Task Specification Trees. Section 5 provides an overview of task planning and its integration with the delegation system. Section 6 then describes the application-specific scan trajectory algorithms in addition to a region partitioning algorithm. Section 7 provides the experimental results generated for the collaborative scanning mission described in the paper.

2 Basic Collaborative Mission Cycle

In any emergency response situation, rescue operators will have access to static interfaces such as ground stations consisting of laptops or stationary PCs in addition to more portable systems such as touch screen devices or smart phones that can be used anywhere in the field. Such devices are set up to provide cognitively efficient multi-modal interfaces to human-robot teams involved in the emergency response. For example, in

the scanning mission described below, the goal is to rapidly provide 3D reconstructions of salient regions by setting up missions through such interfaces.

An operator should be able to request help from its team by simply marking a region on a map in the interface and stating that a 3D re-construction of the region is required. From the operator perspective, it is not important how that is accomplished nor what or how many robotic systems are involved. What is important is that this can be done efficiently within a reasonable span of time and in the right format so that the rest of the team can use the mission output to make better decisions and save lives.

Fig. 1 shows the mission process which begins with an operator specifying a 3D reconstruction mission. Internally, this high-level request is transformed into a *goal request* in the form of a Task Specification Tree (TST, Section 4) representing the goal. This transformation can be achieved dynamically using automated planning techniques, or by using generic TST templates that can be instantiated appropriately.

The setup and execution of the mission represented by the goal request TST should now be *delegated* to one or more participating team members. The TST is therefore given to the local Delegation Module, which initiates a distributed delegation process (Section 4.2) where agents interact through their delegation modules (Fig. 2) using interaction protocols based on speech acts. The process recurses through the tree, filtering potential contractors for each node relative to their capabilities and setting up auctions to determine the platform most fit to be delegated each node. Both mission requirements and platform capabilities can be represented as constraint formulas, and a constraint problem corresponding to the TST and its allocation to agents is incrementally constructed and solved during delegation. Systems may also call other internal functionalities such as motion planners to determine if they can successfully contribute to the goal in question. The net result of the distributed delegation process, if successful, is a new TST representing the collaborative scanning plan resulting from successful delegation. This plan can be sent back to the original human operator for final confirmation, or it can be executed directly as all systems involved have their parts scheduled.

During the mission, the robotic systems involved could stream sparse 3D models to ground stations for real-time visualization or store these models locally for access in a

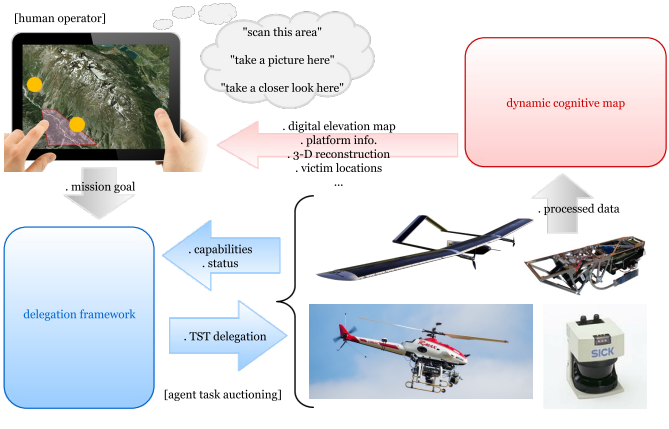

Fig. 1: Mission Process. A goal request is broadcast via the Delegation Module. Platforms with available capabilities reply and a delegation process ensues among each of the platforms' delegation modules. If successful, the net result is a joint plan to execute. Upon execution, raw and processed data can be stored locally or globally. During the mission or upon its completion the human operator can access the results via specialized interfaces.

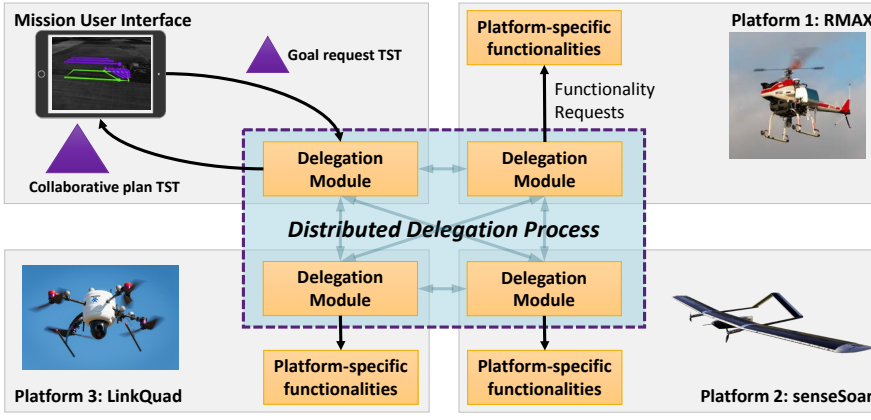

Fig. 2: Overview of the Distributed Delegation Process.

Fig. 3: **(a)** Part of the operational environment of interest depicted in an older photo. **(b)** Sub-region of interest in the second leg of the mission overlayed on an orthophoto generated in the first leg of the mission.

distributed semantic map structure called a Dynamic Cognitive Map (DCM) [2]. The intent is that during the progression of a rescue operation, additional geo-tagged information could be added to this distributed knowledge structure which could be queried by team members for both static and dynamic information related to the operation in progress.

3 A Collaborative 3D Reconstruction Mission

We will now provide additional details regarding the collaborative 3D reconstruction experiment mentioned above. Imagine that an alpine rescue team, consisting of a number of human operators and robotic systems, has been tasked to provide a first-view of an area where some summer hikers may be injured or lost. The first requirement given is to construct a 3D model of the area and then to incrementally refine this model with geographically tagged semantic information. This data and information would be stored in the Dynamic Cognitive Map (DCM) for further use during ongoing rescue missions.

Initially, the team only has a low resolution image of the area, possibly out of date.

Fig. 4: **(a)** The senseSoar solar-powered UAV, developed at ETH Zurich. **(b)** Modular sensor pod for onboard processing and visual-inertial sensing.

The team first wants to get an initial larger overview of the operational environment (Fig. 3a) by rapidly generating a sparse 3D map and orthophotos of this area. The mission operator marks this larger region on a touchpad device and denotes it as R1. This is leg 1 of the mission, which will be executed by the autonomous fixed-wing senseSoar UAV developed at ETH Zurich (Fig. 4). Additionally, the rescue team would like a more detailed 3D map of a subregion dense with interesting physical structures based on the orthophoto generated in leg 1. The mission operator marks the desired region on the touchpad device overlayed on the newly generated orthophoto (Fig. 3b), and denotes this region as R2. This is leg 2 of the mission, which will be executed by an autonomous Yamaha RMAX helicopter developed at Linköping University (Fig. 5).

Below, we describe the robotic systems that will be used in the mission tests in detail:

senseSoar. A hand-launchable, solar-powered, fixed-wing UAV developed in the Autonomous Systems Lab (ASL) at ETH Zurich as a versatile platform for long-endurance sensing and mapping missions (Fig. 4a). The senseSoar is 5kg, with a 3.1m wingspan and an inverted V-tail configuration. Light-weight sensors and avoinics are coupled with a Pixhawk Autopilot for fully autonomous attitude stabilization and waypoint navigation. A modular sensor pod containing forward-oblique and nadir facing visible light cameras, a visual-inertial sensor, and an onboard processing unit is integrated in the fuselage for vision-based 3-D mapping (Fig. 4b).

RMAX. The helicopter platform is based on a slightly modified unmanned RMAX helicopter manufactured by the Yamaha Motor Company (Fig. 5). The RMAX has a rotor diameter of 3.1 meters, a 21 horsepower engine, a maximum take-off weight of 94 kg and a payload capability of about 30 kg. It includes a customized avionics system developed at Linköping University. The avionics box includes two Intel NUC i7 computers, a 2.4GHz WiFi data link, and an Ethernet video server.

The basic sensor suite used for autonomous navigation includes a fiber optic tri-axial gyro system and a tri-axial accelerometer system, an RTK GNSS positioning system and an infrared altimeter used for automatic landing. Additionally, a color and a thermal video camera, as well as a class 1 SICK LMS511 PRO 2D laser scanner, are integrated on board the platform. The laser scanner's maximum range is 80 meters and the maximum scanning field of view is 190°.

In the remaining sections, the generic components in our delegation framework will be described in the context of this mission in order to provide concrete insight into the proposed framework, its theoretical basis and its real-life implementation. We will also describe functionalities specific to 3D reconstruction missions.

Fig. 5: The RMAX helicopter platform with the avionics box attached below the platform and the laser scanner facing downward mounted on a vibration isolated support placed under the helicopter nose. The color and thermal video cameras are mounted on a pan/tilt unit below the avionic box. The GNSS antenna is mounted on the helicopter tail boom.

4 The Delegation Framework

The experimental mission must be *delegated* to suitable participants. Castelfranchi and Falcone [3,4] provide an informal discussion about delegation as a social concept building on a BDI model where agents have beliefs, goals, intentions, and plans [5]. The Delegation Framework discussed in [6] extends these ideas with a formal characterization of delegation in terms of speech acts [7,8] and interaction protocols that implicitly update the belief states of the delegator and contractor. It also supports a concrete delegation *process* using Delegation Modules for each participating agent in a team, resulting in a software architecture for specifying, generating and executing collaborative multi-agent plans to achieve complex goals such as those of the mission above.

The delegation process is based on a recursive algorithm that conceptually sends speech act requests of the type Delegate(*A, B, Task, Context*), where agent *A* wants to delegate *Task* to agent *B* given a *Context* specified as a set of constraints. This could include temporal constraints, restrictions on flight altitudes and velocities, required resolution intervals for various sensors, etc. Agents can be humans or robots. Tasks and missions to delegate are represented as Task Specification Trees (TSTs). While TSTs are declaratively specified, platforms participating in their execution must provide a procedural counterpart capable of interpreting and executing this specification. Internal nodes represent control statements such as sequential and concurrent execution, while leaf nodes can represent domain-specific tasks such as scanning a region.

The current implementation of the Delegation Module (DM, Fig. 6) is based on ROS, Robot Operating System. The TST Factory is used for creating TST nodes and linking them to ancestors and descendants across agents. The TST Executor Factory provides all platform-specific functionality for a node and is consequently instantiated differently for each platform type. It also provides declarative constraints for each node type supported by a platform, representing execution conditions required by the platform itself as opposed to those imposed by a mission. For example, a constraint may define the maximum take-off time for a platform or specify that it can only generate point clouds of a certain resolution. The Delegation Manager is responsible for managing the distributed delegation process and consequently also communicates with other agents. Finally, a constraint solver is used to generate concrete parameter bindings and to verify that all tasks are allocated to platforms capable of executing them.

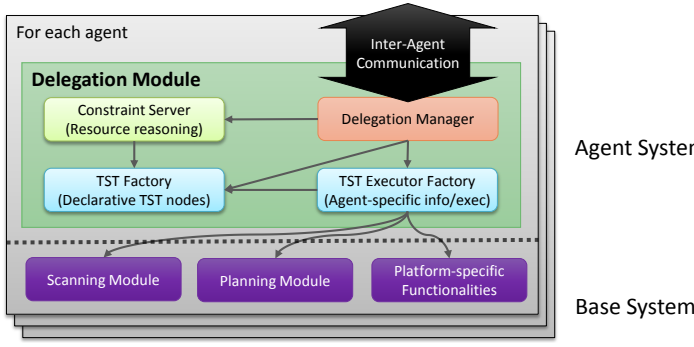

Fig. 6: The Delegation Module

4.1 Task Specification Trees

Task Specification Trees provide a very expressive means of declaratively specifying tasks to be delegated to a team of collaborating agents. Inner tree nodes can specify standardized control structures such as sequences (S), concurrency (C), conditionals (IF) and loops (WHILE), which are directly supported by the Delegation Module. Leaf nodes specify potentially domain-specific tasks to be executed. Such tasks are viewed as elementary and indivisible from the point of view of a delegator, but contractors can choose to elaborate and *expand* them into subtasks through calls to general task planners or problem-specific functionalities during the delegation process.

Every node has a set of named *parameters*, such as the destination of a fly-to task. Parameters can be given specific values or can be subject to *constraints* in a general constraint language [6]. Such constraints represent the GOP's mission requirements.

Mission Example. The delegation process requires a goal request TST. For the 3D reconstruction mission we choose to represent this at a high level of abstraction using scan-map tasks intended to be elaborated with further detail by the agents involved. This allows scan trajectory generation to be adapted to each type of platform in a decentralized manner. Therefore the following small and deceptively simple goal request TST shown in Fig. 7 is generated by the user interface, defining a sequence (S) of scan-map tasks, one for each of the regions (R1 and R2) in the mission. Parameters to these tasks include the desired result (3D point clouds, visual images, object identification, ...), resolution requirements, and requirements for information streaming and storage.

Formal Semantics. TSTs, including control structures and constraints, are given a strict formal semantics through composite actions in TALF, Temporal Action Logic with Fixpoints [9]. Temporal composite actions have the following syntactic form:

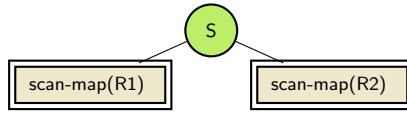

Fig. 7: Goal Request TST

C-ACT ::= $[\tau, \tau']$ with $\bar{x}$ do TASK where ϕ
TASK ::= $[\tau, \tau']$ ELEMENTARY-ACTION-TERM | $[\tau, \tau']$ COMPOSITE-ACTION-TERM |
(C-ACT; C-ACT) | (C-ACT || C-ACT) |
if $[\tau]\,\psi$ then C-ACT else C-ACT | while $[\tau]\,\psi$ do C-ACT

Here, "with $\bar{x}$ do TASK where ϕ" declares a (possibly empty) sequence of variables $\bar{x}$ for use in a constraint formula ϕ related to the given TASK. Such formulas range over all variables in the scope of the formula, which corresponds directly to those parameters and variables that are declared in a given TST node and all of its ancestors. The task must be executed during the given temporal interval $[\tau, \tau']$. For inner nodes, τ and τ' are bounded by constraints relating subtask parameters to subtask durations. The agent executing a task is specified as a parameter to each action term. Before task allocation, this can be an unconstrained agent variable. Note that ELEMENTARY-ACTION-TERM represents a call to an elementary task such as **fly-to**(uav, x, y), whose preconditions, effects and constraints can also be modeled in TALF. COMPOSITE-ACTION-TERM, in contrast, represents a call to a named *composite* action. Semicolon represents sequencing, while || represents potentially concurrent execution.

Task Specification Tree structures can be translated directly to composite actions in TALF, which in turn can be translated to formulas in a fixpoint logic [9] whose expressivity is above that of first-order logic yet allows relatively efficient inference techniques. We emphasize that this translation is mainly used to provide a formal task semantics, and the actual implementation does not require theorem proving.

4.2 The Delegation Process

When a Task Specification Tree has been generated, the user interface can call its local Delegation Manager to initiate the distributed delegation process. This process implements the conceptual Delegate(*A, B, Task, Context*) speech act through an interaction protocol with two phases [6]: First, tasks are provisionally allocated to agents capable of performing them while satisfying all mission constraints. Second, the task allocation and a corresponding constraint solution is accepted or rejected by the operator.

First Phase. The root node of a TST is always a control node and can be handled by any agent. For simplicity we will assume this is delegated to the agent initiating the delegation process. The interaction protocol therefore begins by sending a CALL-FOR-PROPOSAL speech act to this agent [10], indicating the task to be delegated together with the constraint context. From the contractor's point of view, the remainder of the first phase of the protocol can be characterized using the DELEGATE-FIRST-PHASE procedure below. A concrete example will follow.

1: **procedure** DELEGATE-FIRST-PHASE(task T, constraint set C)
2: **if** basic capabilities for root(T) are missing **then reply** REFUSE
3: Add constraints and parameters specified in root(T) to C
4: Add platform-specific constraints for root(T) to C
5: **if** C is inconsistent **then reply** REFUSE
6: **if** root(T) is a leaf and this platform wants to expand it **then**
7: Expand root(T), adding new children

8: **for every** child c_i of root(T) corresponding to a subtree T_i **do**
9: Broadcast a REQUEST to find P = potential contractors with capabilities for c_i
10: Perform auction for c_i among P, and sort P accordingly
11: **nondeterministically choose** $p \in P$:
12: $(T_i', C) \leftarrow p$.DELEGATE-FIRST-PHASE(T_i, C)
13: **replace** T_i with T_i' in T
14: Provisionally commit to the delegation
15: **reply** PROPOSE(T, C)

An agent can only be allocated a tree T if it can execute the root of T. The agent therefore begins by verifying that it has the necessary fundamental capabilities: All agents can coordinate a sequence (S), while only some are able to **fly**. If capabilities are missing, the agent immediately responds using a REFUSE speech act.

The agent must also verify that it can execute the task given the specified parameters and constraints. The currently accumulated set of constraints C is therefore augmented with (1) any mission constraints specific to root(T), corresponding to a where clause, (2) a constraint for each node parameter that was given a specific value outside of the constraints, and (3) any platform-specific execution constraints that this agent has for the given node type, retrieved from the local TST Executor Factory. For example, different UAV platforms have different flight envelopes which must be consistent with mission requirements. If the resulting constraint set is inconsistent, the agent cannot accept the delegation and must reply REFUSE. Otherwise delegation *may* be possible, contingent on the successful delegation of all children. These children may already exist or may be generated dynamically through a potentially platform-specific expansion procedure provided by the TST Executor Factory.

For each child c_i, associated with a subtree T_i, a REQUEST for potential participants will be broadcast. This request is accompanied by a specification of the required capabilities for c_i, which allows replies (sent as INFORM speech acts) to be filtered. An auction process is then initiated where each potential contractor is REQUESTed to bid for the task in question. Each bid is also returned through an INFORM speech act.

Bids are used to prioritize potential contractors, but backtracking may be needed if a choice that is good for one part of the TST has negative consequences for other parts of the tree. For brevity we describe this backtracking using the standard notion of nondeterministic choice, where each such choice point is in fact a point to which the algorithm can backtrack in case of future failures. However, note that agents are called in the order determined by the auction. In this context, failures are reported through REFUSE speech acts, both in the cases discussed above and in case all possible contractors for a child node REFUSE a delegation attempt.

When a child has been provisionally delegated, its subtree may contain expanded nodes, and the nodes of the resulting tree are associated with execution constraints defined by the contractor(s) that were allocated parts of this tree. The expanded tree and updated set of constraints are returned in line 15 and the corresponding values returned from a recursive delegation call are handled in lines 12–13.

When the first phase of delegation succeeds (line 14), the platform also *provisionally* commits to the delegated task before it PROPOSEs a solution to the caller. The commit-

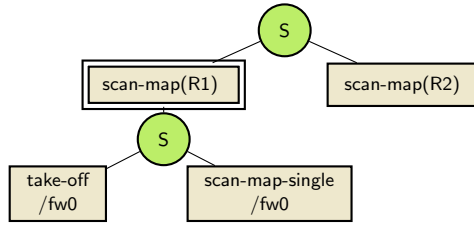

Fig. 8: TST after initial expansion in the senseSoar system.

ment is provisional both because we may backtrack over the commitment and because no delegation is final until the original delegator has received a proposed solution and accepted it. This allows a ground operator to determine whether a mission instantiation is acceptable or whether an alternative needs to be sought.

Second phase. If the mission is accepted, an ACCEPT speech act is distributed to all callers, also specifying a concrete constraint solution to be used during execution. Alternatively, if the mission is rejected, a REJECT speech act is distributed.

Mission Example. In the example, the goal TST is provided as input to the ground operator's delegation module, which starts by attempting to delegate the top node to itself through CALL-FOR-PROPOSAL initiating a call to DELEGATE-FIRST-PHASE.

The delegation process then searches for agents capable of handling the unallocated children of the root. The GOP's delegation module makes a broadcast to acquire all scan-map-capable agents on the team in the GOP's communication range. Two agents respond, the RMAX and the senseSoar. An auction is set up to determine a suitable contractor for scan-map(R1). The cost function for this node type is based on a combination of time requirements and fuel usage. The senseSoar is equipped with visual cameras and scans an area quickly, but the time required for 3D reconstruction rises quickly with the desired resolution. The RMAX uses a laser scanner, and while it flies somewhat more slowly, the results are processed quickly even at higher resolutions. As scan-map(R1) specifies a large area to be scanned, but the resolution required is comparatively low, the senseSoar returns the best bid.

The delegation module therefore first tries to delegate scan-map(R1) to the senseSoar. This node allows a team of platforms to be assigned to the same area and can partition scan regions according to the capabilities of those platforms (Section 6). Partitioning must be performed during delegation rather than execution, to ensure that each platform involved can verify its ability to scan the subregion it is assigned. After this, the scan-map node can expand to specify the concurrent execution of several scanning sub-mission. In this particular mission the team was limited to one member and partitioning was not necessary. Nevertheless the senseSoar expands the node to a sequence of actions suitable for scanning using a single platform, as shown in Fig. 8.

The expanded TST is then recursively delegated. Constraints generated during expansion ensure that the *same* platform will both take off and perform the single platform scanning task. In the general case, generated subtasks can also be delegated to additional agents that can assist in achieving the overall goal of the expanded node.

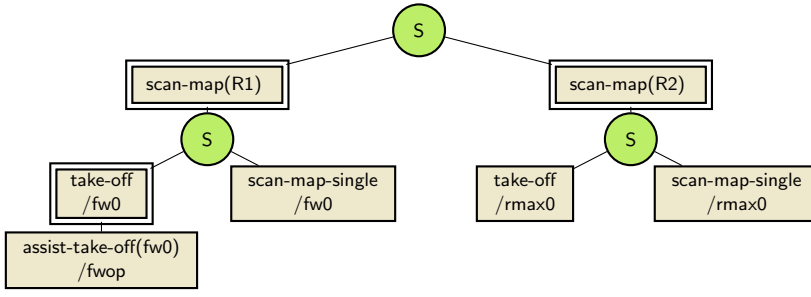

Fig. 9: Fully expanded collaborative plan TST.

While the RMAX can take off autonomously, the light-weight senseSoar requires human assistance. Therefore, when delegated a take-off node, the senseSoar expands this to an assist-take-off node that can only be executed by human agents (included in Fig. 9). When this node is recursively delegated, the auction gives the highest priority to the fixed-wing operator, who has a personal Delegation Module running on a user interface device. Delegation to an operator always asks for confirmation that the human is willing and able to take on the task at the desired time. Modeling take-off as a *potentially* expandable node ensures that only the platform itself needs to know whether it is capable of taking off autonomously, or under which conditions this is possible.

Verifying the ability to scan a region using scan-map-single requires generating a scan trajectory (Section 6.1) and verifying through constraints that this trajectory can be flown in the desired manner. Then, an agent can choose to expand the node into a subtree including a sequence of flight actions or to provide a more complex trajectory-following implementation for the scan-map-single node itself. The latter option results in greater freedom to make choices during execution and was chosen for this mission.

Once the first subtree is successfully provisionally delegated, a similar process ensues for the rightmost leaf node, scan-map(R2). Here the operator desires a high-resolution scan of a smaller area and the auction results in the RMAX being at the top of the list. The GOP's delegation module therefore tries to delegate scan-map(R2) to the RMAX.

As both delegations succeed, the net result is that an expanded *collaborative plan TST* (Fig. 9) where all nodes are allocated to participating agents is proposed to the GOP's delegation module. The GOP can view and approve the plan, after which the approval is transmitted to the participating agents, which then store the TST structures and their associated constraint instantiations and commit to executing the TST at the desired time. Each system will execute its part of the mission TST relative to timing and other constraints that have been checked for consistency in the delegation phase. The results will be shown in Section 7.

5 The Planning Module: Motion and Task Planning

When using unmanned aerial or ground vehicles, *motion planning* is essential for successful execution and is used in several ways. First, when a motion-related TST node

is delegated, the contractor's platform-specific constraints for this task require the existence of a feasible motion plan that is collision-free relative to available obstacle information. A motion planner must then be called to verify feasibility. Moreover, the generated motion plan must be analyzed in order to generate bounds on execution times. For the RMAX, bounds can be generated either through a fast estimation algorithm or through simulated step-by-step execution of the low-level control system. When the TST node is executed, the generated motion plan must be followed.

Though the 3D reconstruction mission was generated through template-based expansion, general *automated task planning* can also be essential for many mission types. Task planning is integrated through the use of a goal node specifying a planning domain and problem instance to be solved. Such goal nodes can occur at any point in a TST, allowing missions to be partly pre-specified and partly generated through planning. These nodes can be handled in one of two ways.

Planning First. As TSTs are sufficiently expressive to represent the output of most automated planners, existing single- or multi-agent planners can easily be integrated. When a goal node is reached, a planner capable of handling the given planning domain language is called and its output is converted into a subtree attached under the goal node. The delegation module then proceeds to recursively delegate the new TST nodes.

Planning Integrated. By adapting a planner to the use of delegation, new actions can be integrated and delegated as soon as they are generated. This has the advantage of immediately testing the feasibility of each action and can therefore reduce the need for backtracking due to plans that cannot be delegated. At the same time, any backtracking caused by the planner itself will trigger backtracking in delegation. Therefore this option is mainly feasible for planners that only cause a limited amount of backtracking. The RMAX uses the planner TFPOP [11,6] for this purpose. TFPOP is a knowledge-rich multi-agent planner whose search algorithm can be guided through additional domain information provided by domain experts. This type of approach has proven orders of magnitude faster than standard planners in many domains, and is particularly appropriate given the need to communicate with other agents when backtracking occurs.

6 Collaborative Scanning and Region Partitioning

Partitioning. Though not used in the example mission, scan-map may specify a team of agents that should collaborate to scan a single region. The contractor coordinates this task, partitioning the region according to the team members' capabilities.

The relative *size* of each subregion can be calculated by determining approximately how large an area can be scanned by each participant in any given period of time. The most significant parameters involved in this calculation can be summarized as follows:

- How quickly can and should the participant fly? This depends on the physical flight envelope of the platform, but also on the characteristics of the sensors being used as well as the desired density and quality of the scan results. For example, a LIDAR sensor may produce a certain number of scan lines per second, and the faster an aircraft flies, the longer the distance between two such lines on the ground. The desired scan quality then restricts the range of permitted air speeds.

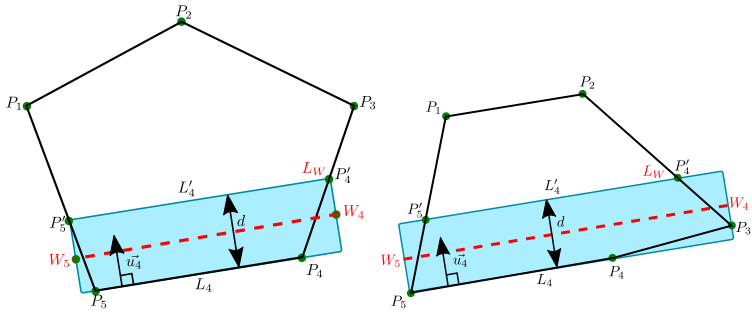

Fig. 10: How to generate one scan strip along the longest edge of a polygon $\mathscr{P}$.

– How wide are the "strips" that can be covered by the sensors in question during a single flight? This depends on field of view restrictions but also on resolution requirements and the range of possible altitudes, which can in turn depend on other mission constraints.

If each team member i can scan an area of a_i per time unit, then it should be assigned a partition whose proportional size is $a_i / \sum_k a_k$ of the area of the entire polygon. We then apply a polygon decomposition algorithm [12] to generate subpolygons of this size, *anchored* in locations calculated from the starting position of each participating UAV.

The expansion of scan-map for multiple participants is very similar to the one shown in Fig. 8, the only difference being that scan-map is followed by a concurrency (C) node whose children consist of one sequence subtree for each platform involved. The scan-map-single node in each such subtree specifies the same mission parameters as the original scan-map node except that the team consists of one specific participant and the scan region is set to its subregion. Consequently each individual member will first take off and then scan its specific partition.

6.1 Scan Trajectory Generation

Scan trajectories can follow patterns such as spirals, expanding squares and lawnmower patterns. For 3D reconstruction, flying in straight lines generally yields more uniform results. It is also important to reduce the number of turns, especially when using fixed-wing aircraft, as turns do not contribute to data collection. We therefore define a trajectory generator that in each step generates a *flight line* along the longest side of the remaining region to be scanned. For rectangles, this results in a lawn mower pattern.

Notation. A polygon $\mathscr{P}$ is defined in terms of a sequence of vertices $\langle P_1 \ldots P_n \rangle \in \mathbb{R}^n$. Fig. 10 shows two examples with $n = 5$ vertices. We introduce the following notation: By $\overline{k}$, we mean $((k-1) \bmod n)+1$. Thus, assuming $n = 5$, we have $P_{\overline{n+1}} = P_1$, ensuring that indexes can "wrap around". L_i is the infinite line going through the points P_i and $P_{\overline{i+1}}$. Fig. 10 shows a finite segment of each line. $d_i = dist(P_i, P_{\overline{i+1}})$ is the linear distance between P_i and $P_{\overline{i+1}}$. For example, d_5 is the distance between P_5 and $P_{\overline{6}} = P_1$.

Algorithm. Trajectories are generated by a recursive algorithm *generate-trajectory*($\mathscr{P}$) that returns a list of flight line coordinates.

(1) Select the scan direction to be parallel to a longest segment L_s of the polygon $\mathscr{P}$, so that $\forall i.d_i \leq d_s$. In both polygons in Fig. 10, line L_4 is the unique longest segment.

(2) Let $\vec{u_s}$ be a vector perpendicular to L_s, pointing toward the interior of the polygon $\mathscr{P}$ (see Fig. 10). Let d be the desired distance between flight lines, taking into account the need for overlap, and let the infinite line L'_s be the translation of L_s in the direction $\vec{u_s}$ by the distance d. Then L_s and L'_s provide two of the edges for the "effective" (non-overlapping) part of a scan strip as highlighted in Fig. 10. Let L_W be the translation of L_s in the direction $\vec{u_s}$ by the distance $\frac{d}{2}$. A finite segment of this line will be followed by the UAV while scanning.

(3) We now want to find two waypoints W_j and W_{j+1} representing the start and end of the new flight line. As illustrated for $W_j = W_4$ and $W_{j+1} = W_5$ in Fig. 10, these can be situated the border of $\mathscr{P}$, to ensure that no part of the polygon is missed.

(3a) Suppose L'_s intersects $\mathscr{P}$ in *two* points, P'_s and $P'_{\overline{s+1}}$, as is the case for L'_4, P'_4 and P'_5 in Fig. 10. We can always find W_{j+1} by considering the "leftmost" point among $P_{\overline{s+1}}$, $P'_{\overline{s+1}}$ and any polygon vertices that may intervene between these two points on the polygon (in this case none). A similar situation applies at P'_s, the other end of L'_s, where P_3 intervenes between P_4 and P'_4 and is the rightmost of these points.

We therefore take the set of all such points, project them orthogonally onto L_s, and select two maximally distant points as W_j and W_{j+1} defining a new flight line $[W_j, W_{j+1}]$. When the flight line is flown, a strip corresponding to the rectangle of width d highlighted in blue will be effectively covered, while information is received from a wider rectangle to ensure overlap and compensate for any lack of precision in following the flight line. The rectangle of width d should now be removed from the polygon $\mathscr{P}$, resulting in a new polygon $\mathscr{P}'$ representing the region that remains to be covered.

$\mathscr{P}'$ is created by removing all the vertices in $\mathscr{P}_{s \to s+1}$, replacing them with P'_s and $P'_{\overline{s+1}}$. The first polygon of Fig. 10 then becomes $\langle P_1, P_2, P_3, P'_4, P'_5 \rangle$, while the second polygon becomes $\langle P_1, P_2, P'_4, P'_5 \rangle$, having fewer vertices than before. Then *generate-trajectory* is called for the new polygon (unless it is empty).

(3b) Suppose instead that L'_s has less than two intersections with the polygon $\mathscr{P}$. Then the remaining part of the polygon will be completely covered after generating this last scan strip. In this case, all vertexes from $\mathscr{P}$ are projected onto L_W, after which the two most distant points are selected as endpoints for the new flight line.

This procedure generates a set of flight lines $\{(W_0, W_1), ...(W_{m-1}, W_m)\}$, which the aircraft can cover in any order. Typically, a helicopter would follow them in the order they are defined, but for a fixed wing it might be better to skip a flight line and then come back to it later in order to accommodate a need for a larger turning radius.

7 Experiment Results: Collaborative 3D Reconstruction

Our experiments took place in Isollaz, Italy, in the region depicted in Fig. 3, at an altitude of 800 meters. One of the missions tested and executed has been described in detail in previous sections of this paper together with the integrated functionalities required to do such a complex mission. For both legs of the mission, the delegation, planning, TST generation, and flight executions were fully autonomous. The intent was to generate 3D models of various resolutions for input into the Dynamic Cognitive

Fig. 11: 3D view of the generated scan pattern for the RMAX and Region 2. The green lines represent the region to scan. The purple lines indicating the lawnmower pattern are marked with differing altitudes.

Map where the models and data could then be used to support alpine rescue teams. Additionally, orthophoto mosaics, individual images and partial semantic classification of the operational environment were also generated from the raw data collected by the two UAVs. The missions tested were part of an evaluation demonstration for the EU project SHERPA (www.sherpa-project.eu).

Since this is an application overview paper, we only summarize the experimental results here, focusing on the integrated framework required for such missions and less on the sensor and fusion aspects. The companion paper [13] presents several collaborative UAV flights from two different locations with diverse terrain (Motala, Sweden and Isollaz, Italy). The distributed reconstructions generated from the optical camera of the fixed-wing and the Lidar of the rotary-wing UAV reveal a relative geo-referencing offset in the range of up to few meters and a slight rotational misfit. Consequently, the companion paper focuses on minimizing the rotational and translational misalignment of the point-clouds. In particular, the performance of classical point-cloud alignment methods such as Iterative Closest Point (ICP) variants [14,15,16,17] are compared to a novel probabilistic data association approach (PDA) [18] that was designed to register dense to sparse point-clouds: In contrast to ICP, PDA associates a point in the source point cloud with a set of points in the target cloud and, in simulation, demonstrated lower misalignment errors [18]. For quantitative results concerning convergence of initial configurations and misalignment errors in the real-world experiments we refer to [13].

Fig. 11 depicts the scan trajectories generated for the RMAX to scan Region 2 in the second leg of the mission. Similar patterns used by the senseSoar and its internal path planner are also generated to scan Region 1 in the first leg of the mission. Each platform uses its own motion planning system. The lawn mower pattern generated by the RMAX is specific to the mission constraints and optimizes paths relative to the sensor constraints of the platform. The objective is to provide 100% coverage of an allocated region at a required resolution. Fig. 12 shows the first strip of the point-cloud generated by the laser scanner mounted on the RMAX. Fig. 13 shows the point-cloud generated by the optical camera of the senseSoar using Pix4D software (pix4d.com).

One of the challenges of these missions is in their distributed nature where different robotic platforms collect different types of data using different sensors with different resolutions for data collection. For such diverse models and data, different fusion and

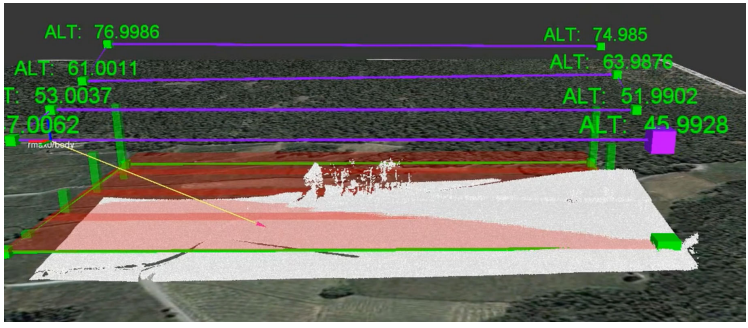

Fig. 12: The first strip of the point-cloud generated by the laser mounted on the RMAX.

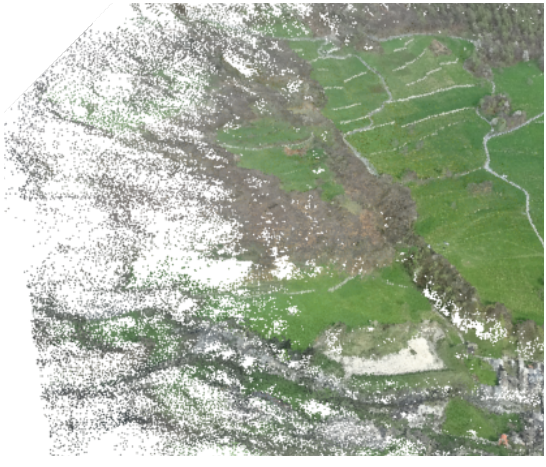

Fig. 13: Part of the point-cloud generated by the RGB camera mounted on the senseSoar using Pix4D software; the points are colored by pixel intensities.

association techniques would be required at many different levels of abstraction to provide a consistent model in the dynamic cognitive map. For instance, Fig. 14 shows a point cloud strip collected by the RMAX. This has to be fused and aligned with the pointcloud generated by the senseSoar from its collection of images, as shown in Fig. 15 which is generated using Pix4D software. Queries now made to the DCM for 3D models of regions could then be done seamlessly, where consistently aligned combinations of low and high resolution data would be output. Once raw data and DEM models are generated and stored in the DCM, one could then begin to build semantically tagged abstractions on top of these 3D models. For instance, using LAStools (http://www.cs.unc.edu/~isenburg/lastools/), the semantic classifications shown in Fig. 16 and Fig. 17 were generated for region 2.

8 Conclusions

This paper has presented a multi-purpose multi-agent/robotic infrastructure that has been deployed and field tested for collaborative 3D mapping applications. It includes a great variety of specific functionalities studied formally in the multi-agent community. The novelty here is that many of these functionalities, although grounded formally, have

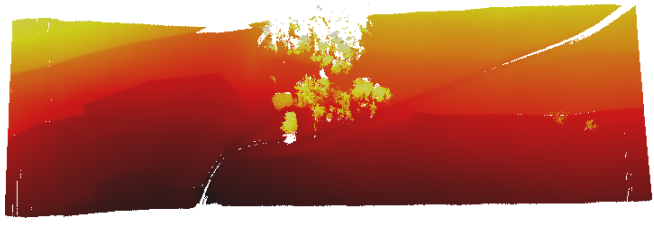

Fig. 14: First Strip of Point-cloud generated by the Laser mounted on the RMAX, including some trees. The pointcloud is colored by height.

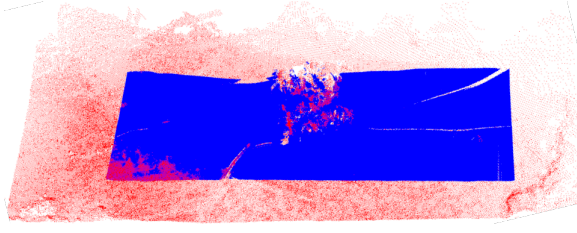

Fig. 15: RMAX strip of point-cloud aligned with the senseSoar point-cloud using Pix4D software.

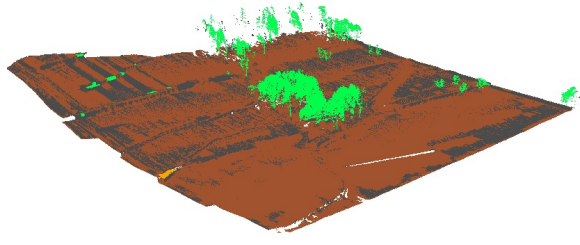

Fig. 16: Color coded semantic classification: Vegetation:Green, Terrain:Brown, Gray:Unclassified

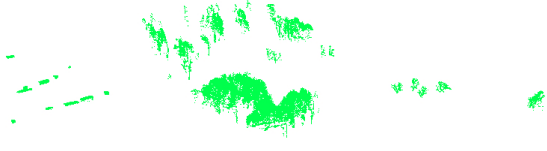

Fig. 17: Filtered semantic classification: Vegetation:Green

been instantiated procedurally in a highly complex, integrated, scalable, collaborative framework for interacting robotic systems. Although only two robots and two humans have been used in the experiment, one can easily add additional robotic systems to a team in a modular and transparent manner. There is much additional research to do in terms of extending the current functionalities and in ensuring the robustness of the infrastructure. The system here can be described as a mature prototype with great potential to push state-of-the-art in multi-agent systems.

Acknowledgments. This work is partially supported by the Swedish Research Council (VR) Linnaeus Center CADICS, the ELLIIT network organization for Information and Communication Technology, the Swedish Foundation for Strategic Research (CUAS Project, SymbiKCloud Project), the EU FP7 project SHERPA (grant agreement 600958), and Vinnova NFFP6 Project 2013-01206.

References

1. R. R. Murphy, "A national initiative in emergency informatics." Computing Community Consortium Version 1: November 3, 2010.
2. L. Marconi and *et al*, "The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2012.
3. C. Castelfranchi and R. Falcone, "Toward a theory of delegation for agent-based systems," in *Robotics and Autonomous Systems*, vol. 24, pp. 141–157, 1998.
4. R. Falcone and C. Castelfranchi, "The human in the loop of a delegated agent: The theory of adjustable social autonomy," *IEEE Transactions on Systems, Man and Cybernetics–Part A: Systems and Humans*, vol. 31, no. 5, pp. 406–418, 2001.
5. P. Cohen and H. Levesque, "Intention is choice with commitment," *Artificial Intelligence*, vol. 42, no. 3, pp. 213–261, 1990.
6. P. Doherty, F. Heintz, and J. Kvarnström, "High-level Mission Specification and Planning for Collaborative Unmanned Aircraft Systems using Delegation," *Unmanned Systems*, vol. 1, no. 1, pp. 75–119, 2013.
7. J. L. Austin, *How to do things with words*. Harvard University Press, 1975.
8. J. R. Searle, *Speech acts: An essay in the philosophy of language*. Cambridge University Press, 1969.
9. P. Doherty, J. Kvarnström, and A. Szalas, "Temporal Composite Actions with Constraints," in *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 478–488, AAAI Press, 2012.
10. FIPA-ACL, *FIPA Communicative Act Library Specification*. Foundation for Intelligent Physical Agents, 2002.
11. J. Kvarnström, "Planning for Loosely Coupled Agents Using Partial Order Forward-Chaining," in *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 138–145, AAAI Press, 2011.
12. S. Hert and V. Lumelsky, "Polygon area decomposition for multiple-robot workspace division," *Int'l Journal of Computational Geometry and Applications*, vol. 8, pp. 437–466, 1998.
13. T. Hinzmann, T. Stastny, G. Conte, P. Doherty, P. Rudol, M. Wzorek, I. Gilitschenski, E. Galceran, and R. Siegwart, "Collaborative 3D reconstruction using heterogeneous unmanned aerial vehicles," in *International Symposium on Robotics*, 2016.
14. P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 239–256, Feb. 1992.
15. Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, pp. 145–155, Apr. 1992.
16. Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int. J. Comput. Vision*, vol. 13, pp. 119–152, Oct. 1994.
17. A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proceedings of Robotics: Science and Systems*, (Seattle, USA), June 2009.
18. G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti, "Point clouds registration with probabilistic data association (submitted)," IEEE, 2016.