FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGICAL CENTER
GRADUATE PROGRAM IN AUTOMATION AND SYSTEMS ENGINEERING

Marcelo Elias Simon

Multi-robots Coordination System for Urban Search and Rescue Post-disaster Assistance Based on Supervisory Control Theory

Florianópolis-SC
2022

Marcelo Elias Simon

Multi-robots Coordination System for Urban Search and Rescue Post-disaster Assistance Based on Supervisory Control Theory

Thesis submitted to the Graduate Program in Automation and Systems Engineering from the Federal University of Santa Catarina for obtaining the Master Degree in Automation and Systems Engineering.
Supervisor: Prof. Max Hering de Queiroz, Dr.
Co-supervisor: Prof. Fabio Luis Baldissera, Dr., Prof. Felipe Gomes de Oliveira Cabral, Dr.

Florianópolis-SC
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Simon, Marcelo Elias
   Multi-robots Coordination System for Urban Search and Rescue Post-disaster Assistance Based on Supervisory Control Theory / Marcelo Elias Simon ; orientador, Max Hering de Queiroz, coorientador, Fabio Luis Baldissera, coorientador, Felipe Gomes de Oliveira Cabral, 2022.
   117 p.

   Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2022.

   Inclui referências.

   1. Engenharia de Automação e Sistemas. 2. Sistema de Coordenação Multi-robôs. 3. Camada Reativa. 4. Camada Deliberativa. 5. Teoria de Controle Supervisório. I. de Queiroz, Max Hering. II. Baldissera, Fabio Luis. III. Cabral, Felipe Gomes de Oliveira IV. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. V. Título.

Marcelo Elias Simon

Multi-robots Coordination System for Urban Search and Rescue Post-disaster Assistance Based on Supervisory Control Theory

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Engenharia de Automação e Sistemas.

ICP-Edu
Documento assinado digitalmente
Werner Kraus Junior
Data: 09/03/2022 15:04:56-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Prof. Werner Kraus Junior, Dr.
Coordenador do Programa de
Pós-Graduação

ICP-Edu
Documento assinado digitalmente
Max Hering de Queiroz
Data: 10/03/2022 08:25:55-0300
CPF: 912.550.079-15
Verifique as assinaturas em https://v.ufsc.br

Prof. Max Hering de Queiroz, Dr.
Orientador

Florianópolis-SC, 2022.

It is with genuine gratitude and warm regard that
I dedicate this work to my parents who gave me
support to accomplish this dream, to my beloved
wife for following me in this path, and for my
brother, sister and all my friends.

ACKNOWLEDGEMENTS

The development of a Master's Thesis is something exciting and enriching, but also arduous. Without some guidance and support it would become harder to be accomplished. So I would like to express my deepest appreciation to my Supervisor Prof. Max Hering de Queiroz, Dr. and Co-supervisors Prof. Fabio Luis Baldissera, Dr. and Prof. Felipe Gomes de Oliveira Cabral, Dr. for their intellectual guidance, motivation and constructive suggestions.

My appreciation also goes for the Coordination for the Improvement of Higher Education Personnel (CAPES) for offering a scholarship as a graduate student, enabling access to higher education. I would also like to extend my deepest gratitude to my parents, Mr. Mauri Simon and Mrs. Laura Salete Müller Simon, thus without their emotional and financial support it would be impossible to go through this experience. Last but not least, I'm extremely grateful to my wife, Mrs. Bianca Rodrigues de Castro, for her patient and enthusiastic encouragement.

RESUMO

A aplicação de robôs em Busca e Resgate Urbano (*USAR - Urban Search and Rescue*) tem recebido maior atenção da comunidade científica por possibiliatar uma redução nos tempos de busca por vítimas e riscos enfrentados por equipes humanas. Entretanto, as tecnologias atuais ainda não proveem a robustez, confiabilidade e segurança necessárias para implementação de robôs em meio a equipes humanas atuando em USAR. Neste trabalho propomos um Sistema de Coordenação Multi-robôs (*MCS - Multi-robots Coordination System*), o qual implementa uma abordagem reativa/deliberativa com o objetivo de fornecer esta tão necessária confiabilidade e segurança. Os robôs e suas restrições foram modelados por modelos formais, para que pudéssemos sintetizar supervisores pela Teoria de Controle Supervisório. Ao implementar esses supervisores no sistema de controle de cada robô, formulamos a camada reativa, a qual delimita ao sistema quais comportamentos atendem às especificações projetadas. Para definir a ação mais eficient dentre as habilitadas pelos supervisores, um Gerenciador de Tarefas (*Task Manager*) é responsável por deliberar o comportamento do robô. Para garantir uma maior eficiência na execução das tarefas solicitadas, um componente centralizado chamado *Task Dispatcher* é então responsável por definir a melhor alocação de tarefas para os robôs com base em seu status atual e sequências de tarefas (missões) requeridas por um humano. A integração total dos componentes do MCS foi implementada pelo *framework* ROS (do inglês, Robotic Operating System) com o objetivo de validar a arquitetura proposta através de um ambiente simulado. Foi projetado um cenário composto por dois robôs terrestres e dois robôs aéreos aplicados a uma cena pós-desastre. Missões foram solicitadas de acordo com os procedimentos comuns de agências USAR internacionais. Como resultado da arquitetura proposta, o MCS fornece maior cofiabilidade aos comportamentos dos robôs, por ser um sistema baseado em formalismos corretos por construção. Além disso, a camada reativa mostrou potencial de ser escalável, apresentando um consumo computacional máximo de apenas 5,7% do sistema de robôs terrestres e 2,3% da capacidade de robôs aéreos. Portanto, o sistema proposto fornece uma alternativa para que robôs sejam aplicados em USAR de forma que suas ações sejam executadas de forma segura.

Palavras-chave: Multi-robôs. USAR. Sistema de Coordenação de Multi-robôs. Camada reativa. Camada deliberativa.

RESUMO EXPANDIDO

INTRODUÇÃO

Desastres urbanos, ocasionados pela ação humana ou por eventos naturais, geralmente resultam em perdas humanas e materiais (MURPHY, 2004b; LIU; NEJAT, 2013). Por exemplo, os ataques ao *World Trade Center* em 11 de setembro de 2001 resultaram em 2.753 mortes, das quais cerca de 400 eram membros de equipes de resgate (CNN, 2019). Para reduzir as baixas humanas desses tipos de desastres, as equipes de Busca e Resgate em Acidentes Urbano (USAR - *Urban Search and Rescue*) devem coordenar eficientemente suas ações em um ambiente perigoso para encontrar e resgatar vítimas o mais rápido possível, pois as chances de sobrevivência das vítimas diminuem exponencialmente ao longo do tempo (MURPHY et al., 2008).

Para mitigar os riscos e dificuldades impostos às equipes de busca e salvamento, pesquisadores vêm investigando a aplicação de robôs móveis em USAR (MURPHY, 2004b; DELMERICO et al., 2019). Os robôs oferecem diversas vantagens, como: podem acessar locais cercados por escombros; suas capacidades de detecção fornecem uma perspectiva útil sobre o cenário global de busca e resgate, aumentando a chamada Consciência Situacional (SA - *Situational Awareness*) de toda a equipe de resgate (LIU; NEJAT, 2013; KLEINER; DORNHEGE, 2007; LIU; NEJAT, 2016); podem operar ininterruptamente; e seu uso permite libertar equipes de resgate de situações ameaçadoras.

Até agora, avanços relevantes foram feitos na robótica USAR, principalmente relacionados a algoritmos de controle de movimento, mapeamento e exploração. No entanto, tópicos relacionados à confiabilidade, robustez e segurança das equipes de robôs ainda precisam de melhorias significativas (DELMERICO et al., 2019).

Uma possível solução para este problema pode ser encontrada no uso de métodos formais aplicáveis a Sistemas a Eventos Discretos (SED). SED são sistemas que têm seu comportamento representado por estados discretos em que a dinâmica é desencadeada por eventos. Assim, podem representar sistemas USAR, uma vez que sua dinâmica não é estritamente dependente do tempo, mas também desencadeada por eventos que ocorrem no ambiente, por exemplo, sinais de sensores. Dentre vários métodos formais propostos pela comunidade SED, a Teoria de Controle Supervisório (TCS) introduzida por Ramadge and Wonham (1987) mostra potencial para o controle de robôs por lidar com ambientes não estruturados e situações desconhecidas como as enfrentadas em cenários USAR. A TCS propõe um método formal para sintetize de controladores minimamente restritivos, garantindo especificações de projeto ao sistema controlado. Considerando os benefícios, a TCS foi selecionada por demonstrar potencial na melhoria da segurança do sistema de controle de robôs, mas também por ser a especialização da equipe da qual o autor é membro.

Portanto, esta Dissertação de Mestrado trata da coordenação segura e confiável de vários robôs heterogêneos executando tarefas simultaneamente em um cenário pós-desastre. Ao estender a arquitetura proposta por Battistella and Queiroz (2014) para um domínio multi-robôs USAR, visamos *atingir especificações de segurança* apesar da ocorrência de falhas modeladas e situações inesperadas, por exemplo, reconhecer um perigo (vazamento de gás) ou uma vítima.

Nesta Dissertação de Mestrado, propomos um Sistema de Coordenação Multi-robôs (SCM) deliberativo e reativo e o validamos em um ambiente simulado. Uma cena pós-desastre desenvolvida e implementada no *freamework* ROS (*Robotic Operating System*) é aplicada para validar o SCM e está disponível em SIMON (2021b). Por meio de casos de estudo, os robôs são obrigados a lidar com situações inesperadas que devem ser superadas com segurança. O sistema se demonstrou capaz de evitar comportamentos inseguros, o que pode aumentar a confiança dos humanos nos robôs autônomos. E embora a TCS seja um assunto complexo, ela foi implementado de forma a simplificar a usabilidade do sistema para humanos.

MULTI-ROBÔS USAR COMO UM PROBLEMA DE CONTROLE SUPERVISÓRIO

O problema de busca e resgate tem, por sua natureza, muitos elementos imprevisíveis. Falhas dos robôs, incertezas impostas pelo ambiente de resgate e a imprevisibilidade da Interface Homem-Robô (IHR) são alguns exemplos.

Quando os robôs são inseridos em ambientes pós-desastres previamente desconhecidos e dinâmicos, alguns problemas relevantes podem surgir, como: garantir maior aplicabilidade (por exemplo, robôs não ficam facilmente indisponíveis devido a falhas e outras situações inesperadas); garantir segurança (por exemplo, as ações de robôs não colocam vidas humanas em perigo); e garantir confiabilidade, o que é muito importante devido ao aumento de sistemas robóticos implementando técnicas de controle não comprovadas matematicamente (por exemplo, algumas abordagens baseadas em inteligência artificial).

Assim, neste trabalho, concentramos nossos esforços no desenvolvimento de uma arquitetura que garanta as seguintes especificações projetadas: 1. não ocorrência de conflitos no uso de sistemas compartilhados; 2. os robôs mantêm os humanos constantemente atualizados sobre as novas descobertas (SA); 3. robôs podem lidar com falhas e ser úteis mesmo com sistemas degradados; e 4. todas as dependências das ações são atendidas antes de sua execução.

Cenário USAR para estudo de caso

Neste trabalho foi proposto o uso de quatro robôs, dois drones (UAV - *Unmanned Aerial Vehicle*) e dois robôs terrestres (UGV - *Unmanned Ground Vehicle*). Estes possuem sensores e câmeras embarcados para tornar possível a sua localização, movimentação e mapeamento de forma autônoma. Também possuem sensores para identificar a presença de vítimas e de gás (apenas para o UGV). E além disto, com base nas missões apresentadas por Murphy (2014) como normalmente atribuídas a robôs, propusemos o conjunto de manobras listadas abaixo:

- *approach* (APP) - Envia o robô para qualquer ponto de interesse;

- *assessment* (ASSESS) - O UAV se move rapidamente sobre o local de trabalho para adquirir informações relevantes;
- *exploration* (EXP) - Exploração total de uma região executada por UGVs com o objetivo de encontrar vítimas e vazamentos de gás;
- *return to base* (RB) - O robô retorna automaticamente para a base de operações;
- *surroundings verification* (VSV) - Representa a ação de olhar ao redor de uma vítima para aumentar SA;
- *victims search* (V_SEARCH) - Uma busca focada, na qual um UAV se aproxima do solo com o objetivo de encontrar vítimas parcialmente enterradas;
- *safe land* (LAND) - Executa um pouso de emergência evitando possíveis vítimas;
- *teleoperation* (TELE) - Modela o comportamento do robô qnd operado diretamente por um humano (teleoperado).

Com os sensores descritos embutidos nos robôs, consideramos UAVs e UGVs compostos pelos seguintes subsistemas: *monitor de consumo de bateria*; *detecção de falhas*; *reconhecimento de vítimas*; *sensor de gás* (somente em UGV); e sistema de *comunicação*.

Como os ambientes USAR são muito imprevisíveis e os robôs não têm conhecimento inicial de sua distribuição, aqui propomos um cenário onde podem ser encontradas as seguintes características: presença de estruturas semi-colapsadas; vítimas distribuídas aleatoriamente; presença de materiais perigosos, por exemplo, vazamento de gás; e ocorrência de eventos inesperados, como falhas dos robôs, níveis baixos de bateria e novas descobertas (vítimas ou perigos).

Teoria de Controle Supervisório

A Teoria do Controle Supervisório é um método formal para a síntese de controladores ótimos formalmente chamados de supervisores que garantem o comportamento do sistema modelado de acordo com as especificações projetadas, apesar da presença de eventos não controláveis (por exemplo, a falha de um robô). Os supervisores são sintetizados de acordo com modelos de planta, que representam o comportamento dos componentes do sistema, e modelos de especificações, que definem as restrições necessárias a serem asseguradas pelo controle supervisório.

Ele pode lidar com situações indesejadas por meio de especificações de segurança, por exemplo, proibindo movimentos do robô devido a falhas do motor que podem tornar seu movimento inseguro. Além disso, garante que o sistema seja não bloqueante, ou seja, o sistema sempre será capaz de atingir um estado objetivo.

Em vez de modelar uma planta global para o sistema físico do robô, nós a dividimos por um conjunto de autômatos que representam cada parte do sistema. Desta forma, considerando os sensores embutidos e manobras executáveis por cada robô, a modelagem resultou em 10 sub-plantas para UGVs e 11 para UAVs.

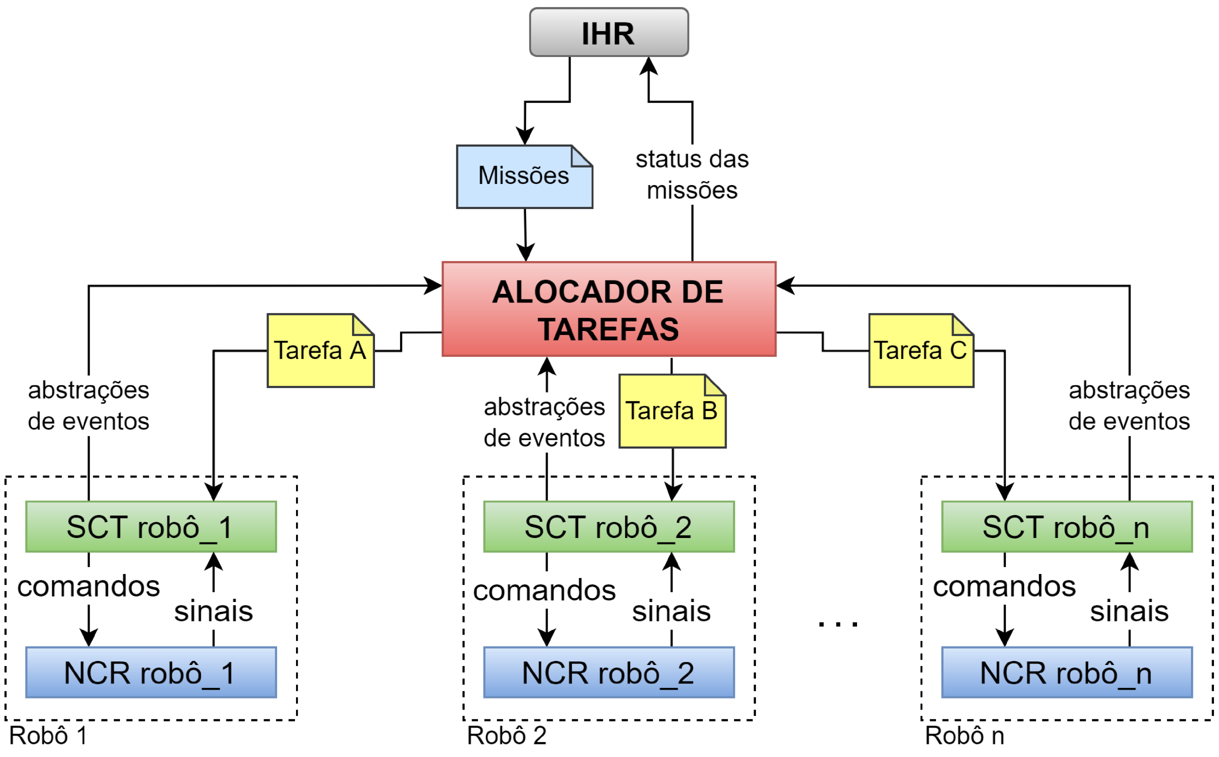
Além dos modelos de plantas, a TCS exige que a modelagem das restrições seja assegurada pelos supervisores sintetizados. Para garantir que os robôs cumpram com as quatro classes de especificações propostas, propusemos 14 modelos para o UGV e 11 para o UAV.

Através de ferramentas computacionais baseadas na TCS, verificou-se que as 11 especificações modulares para os UAVs e as 14 para os UGVs são controláveis em relação à planta, não bloqueantes e não conflitantes entre si. Portanto, a TCS garante que esses modelos de especificação correspondam aos próprios supervisores modulares ideais.

SISTEMA DE COORDENAÇÃO MULTI-ROBÔS (SCM)

A arquitetura proposta neste trabalho é baseada na abordagem de Battistella and Queiroz (2014), que implementa a TCS para um problema envolvendo um robô executando inspeções em barragens hidrelétricas. Os autores implementaram uma arquitetura mista composta principalmente por uma camada reativa responsável por garantir as restrições modeladas ao sistema e uma camada deliberativa projetada para aumentar a eficiência do robô. Como resultado da abordagem, o robô executa as tarefas da forma mais eficiente possível, mantendo sua confiabilidade e segurança. Neste trabalhos implementamos a abordagem de Battistella and Queiroz (2014) em um Sistema de Coordenação de Multi-robôs (SCM), representado pela arquitetura mostrada na Figura 1.

Figura 1 – Visão global da arquitetura proposta. O sistema reativo/deliberativo responsável pela segurança e confiabilidade do sistema está embutido em cada robô, e a coordenação é realizada pelo Alocador de Tarefas, que atribui as tarefas de acordo com heurística proposta.



O sistema proposto é composto por uma IHR, onde o Líder da Força Tarefa pode monitorar os robôs, e atribuir missões, que definem sequências de tarefas a serem executadas pelos robôs. Um **Alocador de Tarefas (AT)** centralizado é responsável por atribuir as tarefas aos robôs e os supervisores sintetizados são incorporados no **Sistema de Controle de Tarefas (SCT)** presente em cada robô. Assim, além da reatividade dos robôs ao ambiente, a interação dos robôs com o AT garante que o AT reaja às mudanças no estado dos robôs realocando as tarefas se necessário.

Camada Centralizada

O Alocador de Tarefas tem a função de extrair as tarefas mais importantes requeridas através da IHR e atribuí-las aos robôs. O Líder da Força Tarefa insere na IHR missões com valores de prioridade de 0 a 10, significando 0 a mais urgente. Cada missão é composta por um conjunto de tarefas, sendo cada tarefa representada pela manobra a ser executada, a região ou posição em que deve ser executada, e o estado requerido da vítima e dos sensores de gás. Após selecionar as tarefas mais importantes o AT seleciona os robôs disponíveis de acordo com seu status que é monitorado através das abstrações de eventos recebidas do Filtro de Eventos (FE) embarcado em cada robô.

Como o objetivo do AT é minimizar o custo necessário para encontrar o melhor robô a ser alocado para cada tarefa, implementamos um algoritmo de árvore de busca. Cada nó é um conjunto de tarefas alocadas aos robôs, e as ramificações representam um novo par robô-tarefa alocado. Para a expansão da árvore, propusemos como heurística a função custo f, apresentada abaixo. Com f pretendemos 1) priorizar tarefas mais urgentes; 2) reduzir a realocação de robôs comprometidos com tarefas de maior prioridade; 3) reduzir a distância percorrida; e 4) reduzir o consumo de bateria de cada robô. Quanto maior o custo, menores as chances de o par robô-tarefa ser selecionado.

$$f(r,t) = pt(r,t) + (10 - pa(r)) + dist(r,t) + (100 - bat(r))/10$$

- $pt(r,t)$: prioridade da tarefa t sendo alocada ao robô r;
- $pa(r,t)$: prioridade da tarefa t atualmente sendo executada pelo robô r;
- $dist(r,t)$: distância euclidiana entre o robô r e o ponto de execução da tarefa t;
- $bat(r)$: nível de bateria do robô r(%);

Camada de Controle Embarcada

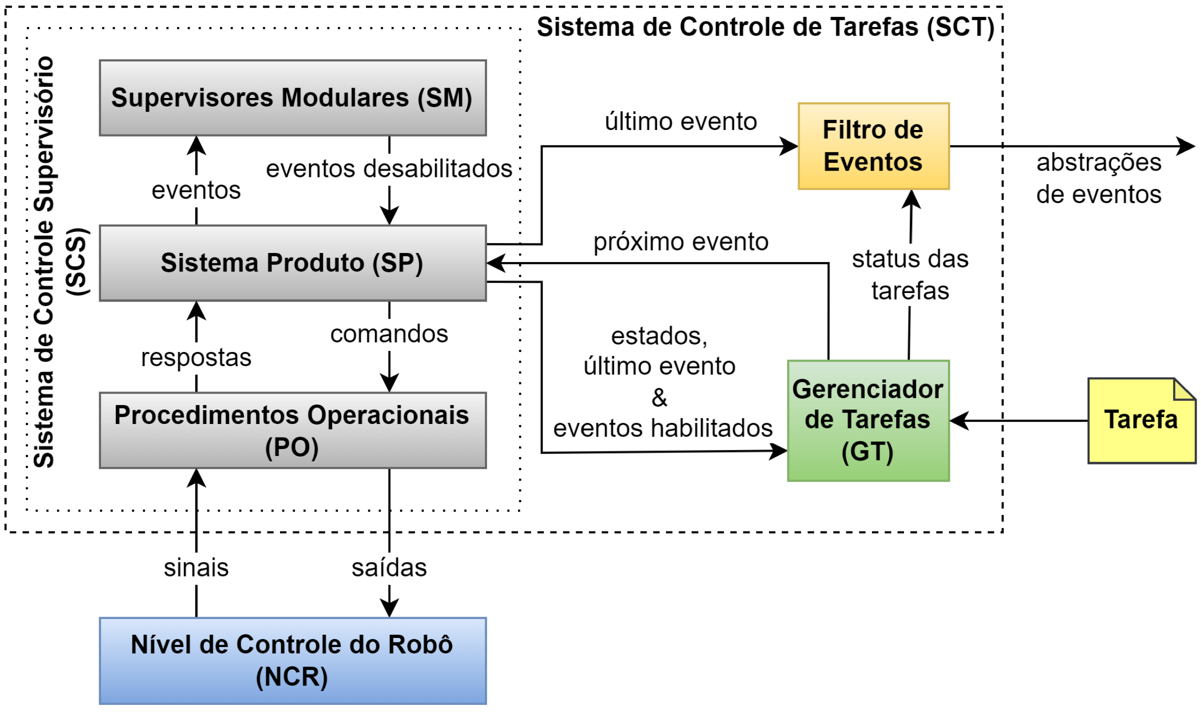
O Sistema de Controle de Tarefas (SCT) mostrado na Figura 2 está embarcado em cada robô com a finalidade de controlar a próxima ação de acordo com a TCS. O SCT é baseado na abordagem do Battistella and Queiroz (2014), que é composto pelo Sistema de Controle Supervisório (SCS) e um Gerenciador de Missões, que aqui denominamos de Gerenciador de Tarefas (GT) uma vez que a atribuição de tarefas é responsabilidade do AT. Para garantir que o AT coordene os robôs, levando em consideração seu status atual, é necessária uma interação eficiente entre o AT e o SCT presente em cada robô.

O SCS, abordado pela primeira vez por Queiroz and Cury (2002b), é uma hierarquia de três níveis que permite a implementação da TCS em sistemas reais. Aqui, modelos e supervisores são implementados através de uma interface entre eventos de alto nível e sinais e comandos de baixo nível reconhecidos pelo Nível de Controle do Robô (NCR), nível no qual todos os sistemas físicos são controlados, por exemplo, sistemas de mapeamento e localização. Essa tradução entre eventos de alto nível e sinais de baixo nível é realizada pelo bloco **Procedimentos Operacionais (PO)**. Podendo ser uma conversão de um valor analógico para uma representação discreta (por exemplo, porcentagem de alimentação da bateria para níveis discretos), ou uma conversão direta de um evento para um sinal físico ou sequência de sinais.

Para controlar os robôs, as plantas modeladas são implementadas no **Sistema Produto (SP)** e sua dinâmica é definida pelos eventos forçados pelo Gerenciador de Tarefas e pelas respostas provenientes do PO. Ao transmitir os eventos ocorridos e receber os eventos desabilitados, o SP também interage com os **Supervisores Modulares (SM)**, que implementam os supervisores sintetizados. Por exemplo, uma mudança no nível da bateria de 10% para 9% levaria o sistema a um estado onde a bateria é considerada crítica, resultando em supervisores restringindo alguns comportamentos do robô.

Propomos um bloco chamado **Filtro de Eventos (FE)**, responsável por traduzir alter-

Figura 2 – Os elementos do SCT interagem entre si para garantir o comportamento do robô de acordo com as especificações modeladas e fornecer a interação necessária com o AT através de eventos abstraídos. Adaptado de Battistella and Queiroz (2014).

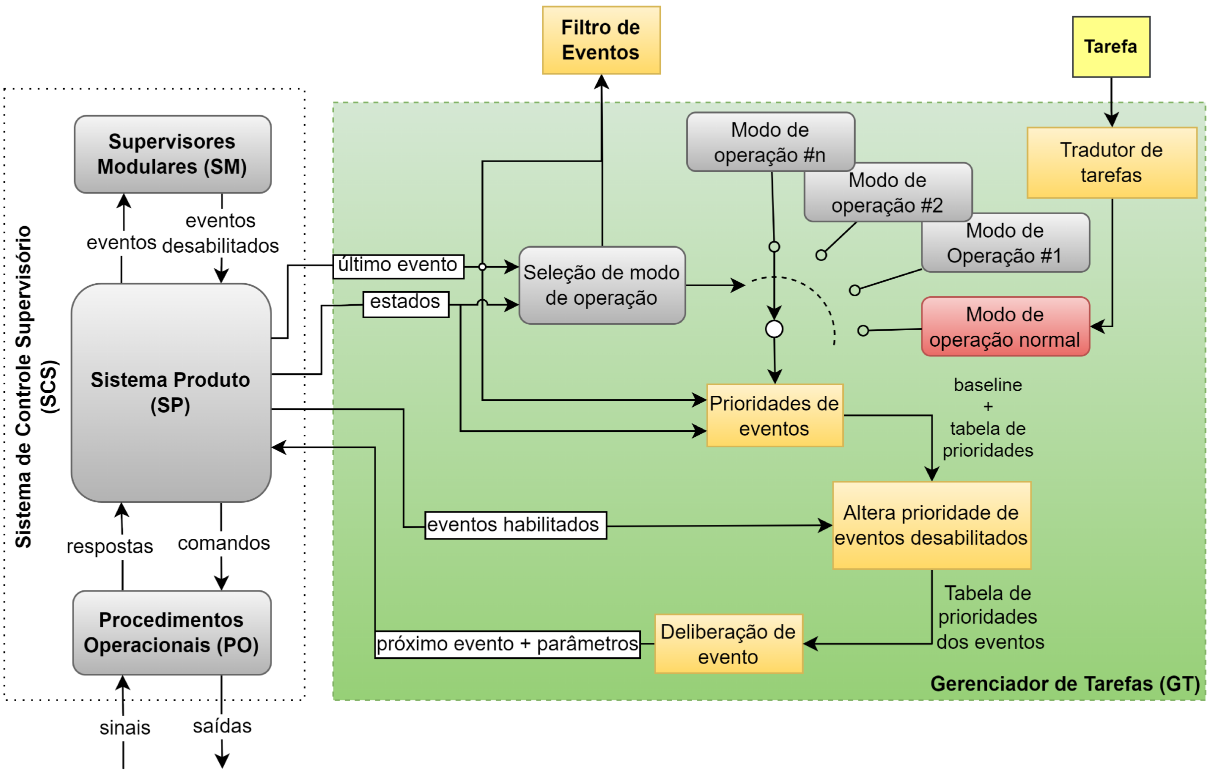

Sistema de Controle de Tarefas (SCT)
Sistema de Controle Supervisório (SCS)
Supervisores Modulares (SM)
eventos
eventos desabilitados
Sistema Produto (SP)
respostas
comandos
Procedimentos Operacionais (PO)
sinais
saídas
Nível de Controle do Robô (NCR)
último evento
Filtro de Eventos
abstrações de eventos
próximo evento
status das tarefas
estados, último evento & eventos habilitados
Gerenciador de Tarefas (GT)
Tarefa

ações no estado do robô que sejam importantes para a seleção de robôs executada pelo AT. O envio de todos os eventos exigiria que o AT implementasse todas as plantas de robôs e aumentaria o número de mensagens trocadas. Assim, o FE monitora a dinâmica do robô e abstrai as mudanças em um conjunto menor de eventos. Como a função de custo implementada no AT depende do status das tarefas e a seleção dos robôs é baseada no status dos robôs e sensores, o FE monitora: erros dos sensores; mudanças de status do robô; e alterações de tarefas devido a um novo modo de operação selecionado ou finalização de tarefas. No exemplo descrito, o nível crítico da bateria, assim como uma falha crítica, seria traduzido pelo FE como um evento 'robot_unable', fazendo com que o AT soubesse que o robô não pode receber novas tarefas.

O **Gerenciador de Tarefas (GT)** mostrado na Figura 3 também é muito importante para a implementação do SCT e apresenta componentes reativos e deliberativos. Ele é responsável por selecionar o modo de operação do robô em reação aos eventos reconhecidos pelo SCS e novas tarefas recebidas. Cada robô foi projetado com cinco modos de operação e um modo de operação normal. Três deles são modos de operação degradados em que o robô reduz o conjunto de manobras possíveis ou executa um comportamento de backup, por exemplo, UAVs executam um pouso de emergência devido ao nível crítico da bateria. Os outros dois modos alteram o comportamento do robô para realizar a verificação do entorno de uma vítima ou a teleoperação.

Quando há uma mudança no PS ou uma nova tarefa chega, o GT seleciona o modo

Figura 3 – O Gerenciador de Tarefas controla o comportamento do robô com base em seu estado atual, eventos habilitados pelos supervisores e tarefas atribuídas pelo AT. Adaptado de Battistella and Queiroz (2014).


Filtro de Eventos
Tarefa
Sistema de Controle Supervisório (SCS)
Supervisores Modulares (SM)
eventos
eventos desabilitados
Sistema Produto (SP)
respostas
comandos
Procedimentos Operacionais (PO)
sinais
saídas
último evento
estados
Seleção de modo de operação
Modo de operação #n
Modo de operação #2
Modo de Operação #1
Modo de operação normal
Tradutor de tarefas
Prioridades de eventos
baseline + tabela de prioridades
eventos habilitados
Altera prioridade de eventos desabilitados
Tabela de prioridades dos eventos
próximo evento + parâmetros
Deliberação de evento
Gerenciador de Tarefas (GT)

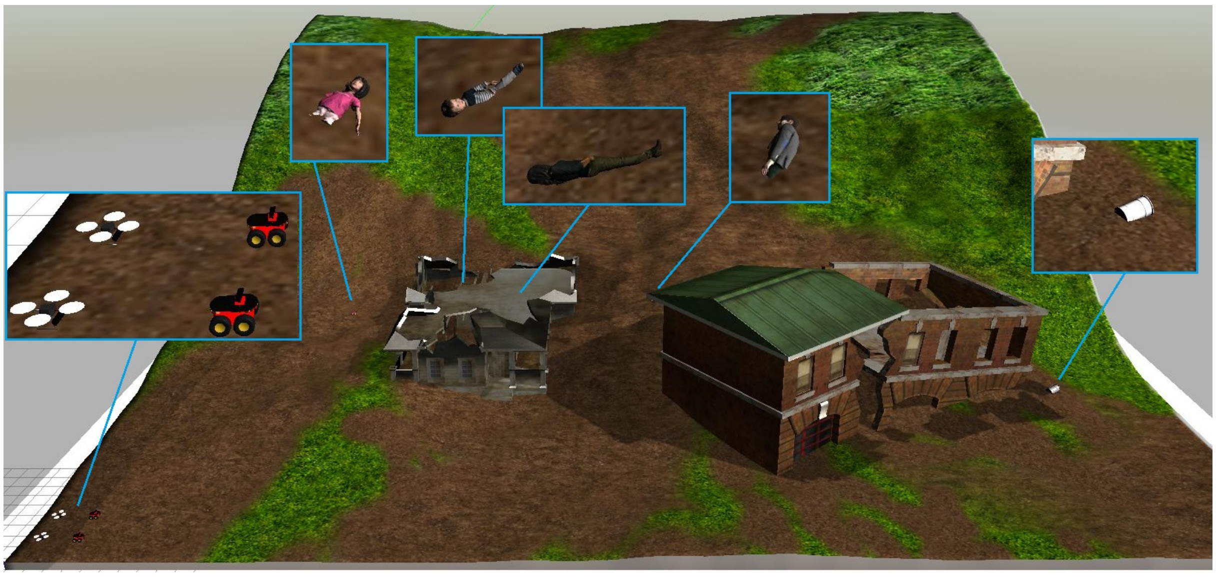
de operação mais adequado e atribui prioridades a todos os eventos modelados. De acordo com as prioridades atribuídas a cada evento delibera se aguarda a ocorrência de um evento não controlável ou se aciona um novo controlável.

RESULTADOS E DISCUSSÕES

Para validar que o SCM garante a rápida realocação de tarefas, o comprometimento dos robôs com as especificações modeladas e a escalabilidade do sistema, implementamos um ambiente simulado. O cenário é o mostrado na Figura 4 e é composto por dois prédios semi-colapsados, quatro vítimas e duas fontes de vazamento de gás, sendo que dois UGVs e dois UAVs são responsáveis pela execução das tarefas exigidas pelo Líder da Força Tarefa. O SCS foi implementado em Python com o auxílio de um framework dedicado disponível em SIMON (2021a) e o sistema completo composto pelo ambiente simulado e robôs controlados pelo SCM está disponível em SIMON (2021b).

Nesse cenário, robôs foram obrigados a realizar a avaliação de áreas amplas, rodear os prédios e vasculhá-los e, posteriormente, as regiões compostas pelas duas vítimas externas tiveram sua exploração solicitada. Enquanto os robôs executavam as tarefas atribuídas pelo AT nós forçamos eventos inesperados como a falha do sensor de gás de um UGV, falhas simples e críticas, erros de manobras e atribuição de novas tarefas.

Figura 4 – Ambiente USAR simulado e respectiva distribuição de edifícios, vítimas e perigos. Duas das vítimas foram colocadas do lado de fora e as outras duas foram colocadas no prédio à esquerda. Os locais perigosos foram representados por dois pontos com vazamento de gás próximo ao prédio à direita.



Resposta do AT à alterações ocorridas nos robôs

Inicialmente o AT atribui aos robôs terrestres as missões de circular os edifícios e explorar a área onde estão contidos. Na Figura 5 apresentamos o resultado da interação entre o AT e o EF embarcado nos robôs. Quando o UGV2 reconhece uma falha no sensor de gás ele informa imediatamente o AT (através do FE), que reatribui as tarefas para fazer com que ambas as UGVs troquem seus objetivos. Isso acontece porque a missão relacionada ao prédio da direita exige que o sensor de gás esteja ligado, enquanto que no da esquerda não foi necessário.

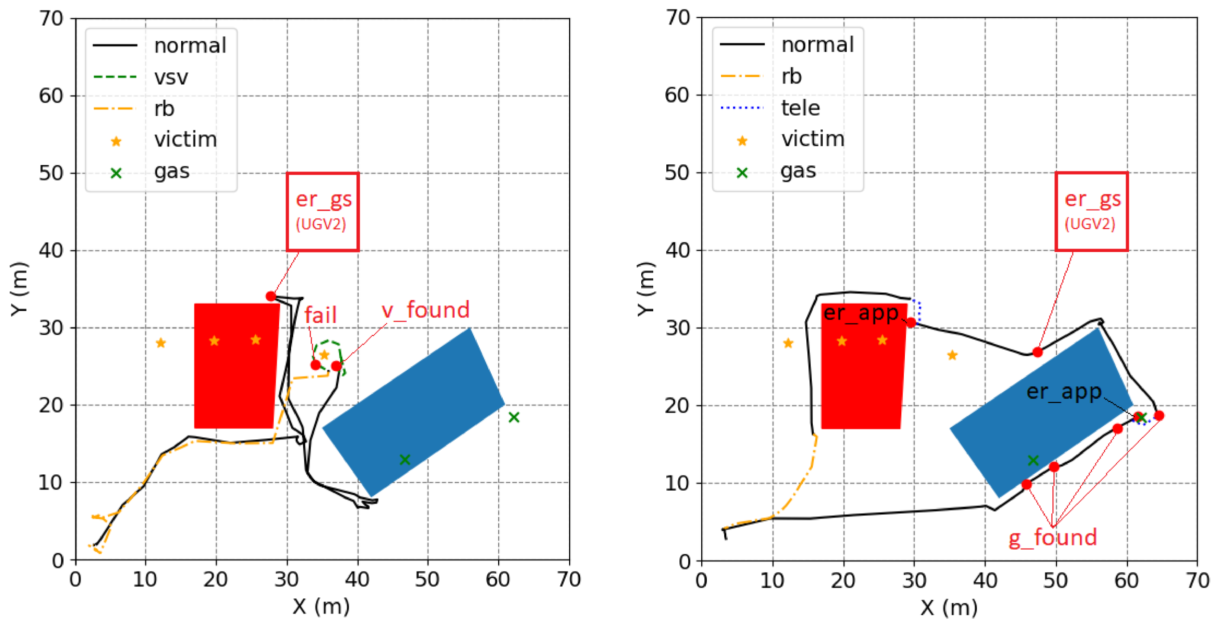
Compromisso dos robôs com as especificações modeladas

Os robôs começam na base de operações sem tarefas iniciais atribuídas a eles. Esta base é colocada no canto esquerdo para baixo e é a posição que os robôs tentarão ir quando a manobra de retorno à base for executada. Tal comportamento pode ser observado na Figura 5 pela linha tracejada amarela, demonstrando que ambos os robôs retornaram à base. A UGV2 executou tal comportamento por requisição do operador humano e o UGV1 devido a uma falha simples ocorrida em seu sistema.

Como mencionado, o GT implementa comportamentos de backup que são selecionados de acordo com o estado dos robôs. Quando um UGV identifica uma vítima, ele muda para um comportamento onde as tarefas atuais são suspensas enquanto a verificação do entorno da vítima é executada. Na Figura 5a é mostrado este comportamento executado pelo UGV1, que cerca a vítima. O UAV1 também encontra uma vítima e como é mostrado na Figura 6 executou um movimento em espiral em torno da vítima para verificar a possibilidade de encontrar outras vítimas próximas à primeira.

Um dos modos de operação degradados implementados é acionado pelas falhas simples ou pelo baixo nível de bateria (bat_L). Ambos os tipos de robôs executam o retorno

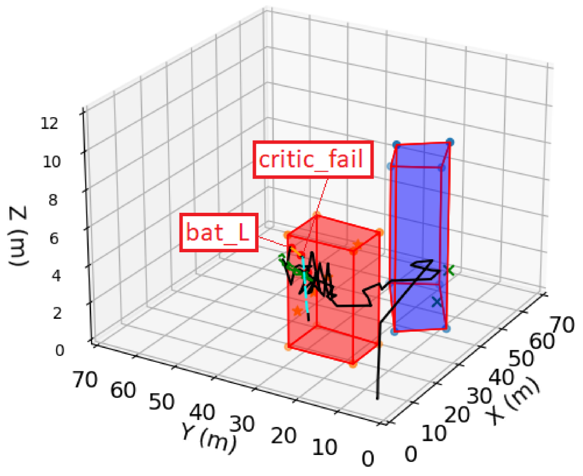
Figura 5 – Trajetórias de robôs terrestres. Devido a uma falha no sensor de gás o AT troca as tarefas dos robôs para que ambas as missões possam ser cumpridas.



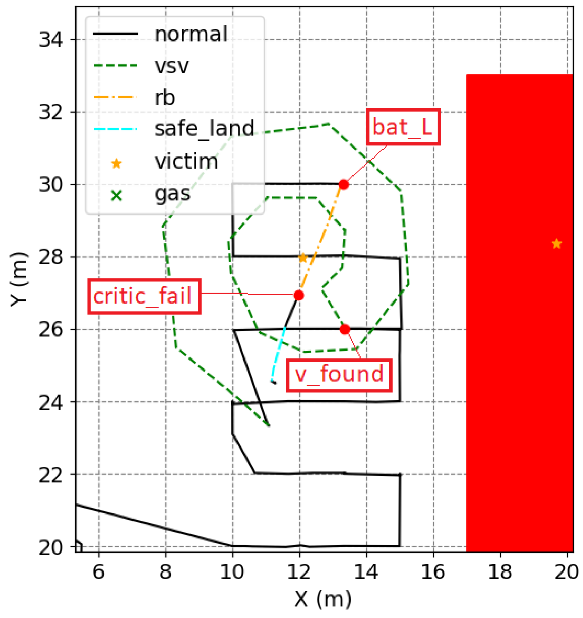
(a) UGV1.

(b) UGV2.

Figura 6 – Trajetória do UAV1 no ambiente simulado.



(a) Trajetória do UAV1.



(b) Zoom da vista superior da trajetória do UAV1.

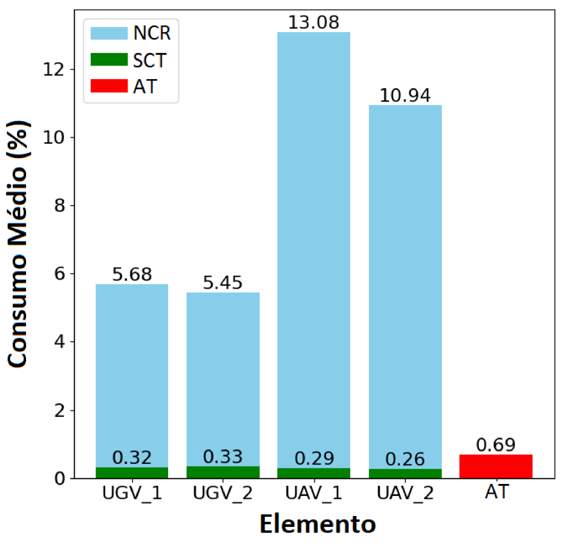
à base quando tal comportamento é acionado, mas com uma pequena diferença. O UAV o executa imediatamente e aborta a tarefa atual, e o UGV executa o retorno somente após terminar a manobra atual em execução. Na Figura 5a o UGV1 reconhece a falha simples (fail) durante a execução do VSV e continua a manobra até o seu término, desde que permitido pelos modelos.

Escalabilidade do SCM

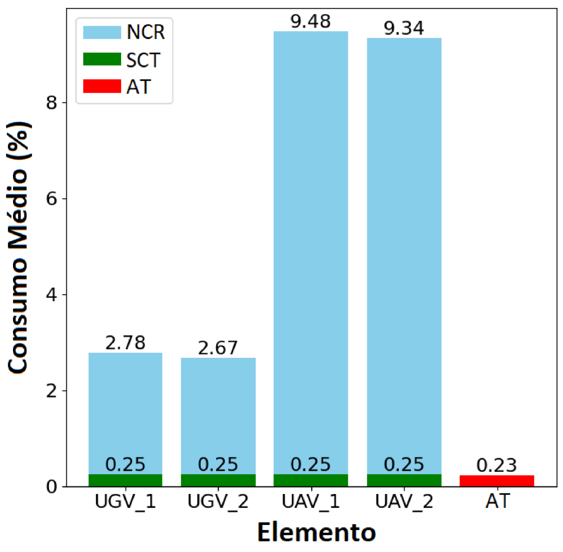
Além de analisar o comportamento dos robôs em diferentes situações inesperadas, também avaliamos o uso de memória e CPU do sistema implementado. Todas as simulações foram implementadas em uma Máquina Virtual (MV) fornecida pelo *Google Cloud Platform*. A MV implementada foi uma c2-standard-8 composta por um processador Cascade Lake com 8 núcleos e 32GB de memória RAM, e Ubuntu 18.04 como sistema operacional.

Na Figura 7, mostramos a porcentagem média de uso de CPU e RAM das camadas SCT e RCL dos robôs, bem como do AT. Enquanto a camada SCT dos robôs consumiu uma quantidade insignificante de CPU (cerca de 0,3%), o nível de controle do robô exigiu mais esforço (até 5,68% para UGVs e 13,08% para UAVs). A mesma baixa influência é verificada no consumo de memória, o que nos faz acreditar que a implementação do SCT nos robôs não é uma limitação para a escalabilidade dos robôs.

Figura 7 – Custo computacional da arquitetura proposta. A implementação do SCT e AT tem um impacto muito baixo no custo computacional quando comparado ao esforço exigido pelo NCR.



(a) Uso da CPU.



(b) Uso da MEMÓRIA.

CONSIDERAÇÕES FINAIS

A necessidade de sistemas robóticos mais confiáveis e seguros se mostra como um passo crucial para a real implementação de robôs no domínio USAR. Assim, apresentamos neste trabalho uma arquitetura deliberativa-reativa que permite que robôs executem diferentes tarefas de forma coordenada, garantindo segurança para si e para os humanos envolvidos. A abordagem baseia-se na implementação da Teoria de Controle Supervisório para controlar os sistemas dos robôs de forma confiável enquanto um sistema centralizado é responsável por coordenar as tarefas que precisam ser executadas por cada robô.

Cada robô pode assumir diferentes modos de operação de acordo com o estado atual dos modelos projetados. Com o SCS e GT embarcados é possível garantir que os robôs atendem as especificações modeladas, aumentando a confiabilidade das ações executadas. Ao mesmo tempo, a integração entre o AT e o SCT possibilitou o rápido replanejamento das tarefas como reação às mudanças no status dos robôs.

Os modelos propostos não cobrem todos os aspectos do sistema de robôs, mas representam os componentes básicos e podem servir de base para futuras implementações. Para motivar ainda mais a aplicação da arquitetura proposta, sua implementação completa está disponível em SIMON (2021b), onde é possível inserir diferentes sistemas de baixo nível e verificar seu comportamento ao ser controlados pelas camadas de alto nível propostas.

O desenvolvimento deste trabalho e os resultados obtidos a partir do ambiente simulado mostraram-se relevantes e podem ser considerados um passo em frente na implementação real de robôs mais autônomos no campo USAR. No entanto, questões não resolvidas ainda podem ser abordadas em trabalhos futuros. Por exemplo, implementar um número maior de robôs para validar a escalabilidade do sistema; atualizar a arquitetura para uma não centralizada (por exemplo, baseada em multiagentes ou comportamentos de enxame); ou inserir especificações que restringem o comportamento de um robô ao estado dos outros.

ABSTRACT

The application of robots in Urban Search and Rescue (USAR) has received an increasing attention from the scientific community to reduce the time needed to find victims and risks faced by human teams. However, current technologies still do not provide the robustness, reliability and security required for implementing robots as part of human teams working in USAR. In this work we propose a Multi-robots Coordination System (MCS) which implements a reactive/deliberative approach in order to provide this much-needed reliability and security. The robots and their constraints were modeled by formal models, so that we could synthesize supervisors by the Supervisory Control Theory (SCT). By implementing these supervisors in the control system of each robot, we formulate the reactive layer, which delimits to the system which behaviors attain to the designed specifications. To define the most efficient action among those enabled by supervisors, a Task Manager is responsible for deliberating the robot's behavior. To ensure greater efficiency in executing the requested tasks, a centralized component called Task Dispatcher is then responsible for defining the best task allocation for robots based on their current status and task sequences (missions) required by a robot. human. The total integration of MCS components was implemented in the Robotic Operating System (ROS) framework in order to validate the proposed architecture through a simulated environment. A scenario consisting of two ground robots and two aerial robots was designed and applied to a post-disaster scene. Missions were requested according to the common procedures of international USAR agencies. As a result of the proposed architecture, the MCS provides greater reliability to the behavior of robots, as it is a system based on correct by construction formalisms. In addition, the reactive layer showed potential to be scalable, presenting a maximum computational consumption of only 5.7% of the ground robot system and 2.3% of the capacity of aerial robots. Therefore, the proposed system provides an alternative for robots to be applied in USAR so that their actions are performed safely.

Keywords: Multi-robots. USAR. Multi-robots Coordination System. Deliberative Layer. Reactive Layer.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

DES	Discrete Events Systems
EF	Events Filter
FEMA	Federal Emergency Management Agency
GCP	Google Cloud Platform
GUI	Graphical User Interfaces
HRI	Human Robot Interface
INSARAG	International Search and Rescue Advisory Group
LMSCT	Local Modular Supervisory Control Theory
MCS	Multi-robots Coordination System
MS	Modular Supervisors
OP	Operational Procedures
PS	Product System
RCL	Robot Control Level
RL	Reinforcement Learning
ROS	Robotic Operating System
SCS	Supervisory Control System
SCT	Supervisory Control Theory
SLAM	Simultaneous Localization and Mapping
TCS	Task Control System
TD	Task Dispatcher
TM	Task Manager
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USAR	Urban Search and Rescue
VM	Virtual Machine
WTC	World Trade Center

CONTENTS

1 INTRODUCTION

Urban disasters, induced by human action or by natural events, generally result in both human and material losses (MURPHY, 2004b; LIU; NEJAT, 2013). For example, the attacks to the World Trade Center (WTC) on September 11th, 2001 resulted in 2,753 deaths, of whom about 400 were members of rescue teams (MURPHY, 2004b; CNN, 2019). As another example, the series of earthquakes that occurred in Haiti in the year 2010 resulted in a death toll of more than 300,000 people in a disaster that extended more than 200 km from the epicenter. It also resulted on the total destruction of 105,000 homes (REPUBLIC OF HAITI, 2010).

To reduce human casualties from these types of disasters, Urban Search and Rescue (USAR) teams must efficiently coordinate their actions to find and rescue victims as fast as possible. As victims' chances of survival decrease exponentially over time, the pressure to find them makes rescue teams work with almost no rest time. The unstructured environment imposes difficulties on the search for victims. Dangerous situations may arise at any moment, delaying the team response to ensure their safety, e.g., discovery of a point with gas leakage. Under these conditions, the attention of the team starts to degrade over the days and it becomes difficult to perform in an efficient way (MURPHY et al., 2008).

Aiming to mitigate the risks and difficulties imposed on search and rescue teams, researchers have been investigating the application of mobile robots in USAR (MURPHY, 2004b; DELMERICO et al., 2019). Robots offer several advantages, such as: they can access places surrounded by rubble; their sensing capabilities provide an useful perspective on the global search and rescue scenario, increasing the so-called Situational Awareness (SA) of the whole rescue team (LIU; NEJAT, 2013; KLEINER; DORNHEGE, 2007; LIU; NEJAT, 2016); they can operate uninterruptedly, since robots can be intercalated; and their use allows rescue teams to be freed from threatening situations.

Until now, relevant advances have been made in USAR robotics, mostly related to motion control, localization, mapping and exploration algorithms. However, the reliability and safety of the actions executed by the robots in USAR environments, and robustness of their control system have still not received relevant attention from the community (DELMERICO et al., 2019). In this work, read "robust control system" as a high-level system that ensures the correct behavior of the robot, even with the implementation of different low-level control systems. Considering this gap pointed out by Delmerico et al. (2019), we focus our efforts on the robust, safe and reliable control of multiple heterogeneous robots and their coordination while simultaneously executing tasks in a post-disaster scenario. Aiming *to attain safety specifications* despite the occurrence of modeled failures and unexpected situations, e.g., recognising a danger (gas leakage) or a victim.

Furthermore, when designing, developing and implementing autonomuous technologies, they must follow ethical concepts related to human-to-robot interaction, whether direct or indirect. Thus, the design of a multi-robots system must follow the following general ethics principles: 1. the system must adhere to international recognized **Human Rights**; 2. it must prioritize metrics of **Well-being**; 3. it have to ensure the **Accountability** of designers and operators; 4. it must ensure **Transparency** of the operation; and 5. it must minimize the **risks of being misused** (DESIGN, 2019).

The search and rescue problem has, by its nature, many unpredictable elements. Failures of the robots, uncertainties imposed by the rescue environment, and the unpredictability of the Human-Robot Interface (HRI) are some examples. Formal methods are appropriate for such problems because they guarantee the correct behavior of the system by mathematically proven approaches. Some use of these methods in robotic systems will be further presented in Chapter 3.

For example, the use of Petri Nets and Model Checking techniques have been applied to robotic systems to solve collaboration/cooperation problems (ZIPARO, Vittorio A et al., 2011; COSTELHA; LIMA, P., 2012), path planning (KOO et al., 2012; KLOETZER; MAHULEA, 2019), and also as a solution to reduce exploration cost (DAI; JIANG; LI, 2019). Some formalisms have also been applied to the USAR domain, Carbone et al. (2008) and Talamadupula et al. (2010) use formal methods to determine how a robot reacts to the environment since it is very unpredictable. The implemented methods can increase the reliability of robots, ensure correct reaction to changes in the environment, and also determine the best plan that guarantees modeled characteristics. However, differently from Supervisory Control Theory (SCT), they do not synthesize the controllers by a formal process and, sometimes, this may result in too restrictive solutions.

The Supervisory Control Theory proposed by Ramadge and Wonham (1987) is a formal method that consists of a procedure to synthesize controllers from Discrete Event Systems (DES). This theory, explained in Chapter 4, is characterized by the synthesis of supervisors that guarantees project specifications despite the presence of non-controllable events, e.g., the failure of a robot. The SCT can handle undesired situations through safety specifications, e.g., prohibited conditions for robots. In addition, it is able to guarantee the liveness of the system, e.g., avoid system blocking situations. Thus, the implementation of a high-level system based on the Supervisory Control Theory might be well suited to the problem being faced since it can deal with the unstructured environment and unknown situations faced in USAR scenarios. Nevertheless, the state space explosion could be a limitation to the scalability of robots applied to the system.

Battistella (2015) introduces a reliable approach based on the SCT applied to a problem involving a robot executing inspections on hydroelectric dams. A mixed architecture mainly composed of deliberative and reactive layers is proposed. The SCT-based reactive layer ensures that only safe sequences of events can be executed, ensuring robustness and safety for the behavior executed by the robot. The deliberative layer, on the other hand, makes it possible to plan the best sequence of events between the enabled ones. It results in a robot executing tasks as efficiently as possible while maintaining its reliability and safety.

In this Master Thesis, we verify if the SCT fits well with the USAR problem by implementing a variation of the approach of Battistella (2015) directly in the control system embedded on each robot. Besides, a centralized system interfaces with humans, who define tasks to be accomplished, and with the robots, with the objective of allocating to them the tasks of higher priority. The SCT was selected due to its benefits and potential in improving the safety of the control system of robots, but also because it is the specialization of the team of which the author is a member. Here we highlight that the SCT is only implemented in the internal control systems of each robot and is not applied to the centralized allocation system, which only monitors the occurrence of relevant events in robots to accomplish the allocation based on an A* search algorithm. Thus, here we propose an architecture for the USAR multi-robots domain in a way to solve the following aim and objectives.

1.1 AIMS AND OBJECTIVES

1.1.1 Aim

The aim of this work is to exploit the SCT to design a control system to be embedded in heterogeneous robots applied in USAR, ensuring robust, safe and reliable individual behaviors when integrating a team of robots.

1.1.2 Objectives

To successfully achieve the aim of this Master Thesis, the following objectives have been proposed:

1. Define a representative search and rescue scenario that simulates an USAR environment, focused to the validation of the proposed control system;

2. Define formal models for robots behaviors and payloads, specify safety conditions from the USAR perspective, and formulate the use of multi-robots in USAR as a supervisory control synthesis problem. And by this, explore the benefits ensured by the SCT (liveness, maximum permissiveness, correct by construction controllers);

3. Propose a multi-robots coordination system for USAR situations, capable of allocating tasks to multiple robots with an embedded control system that extends the deliberative-reactive approach developed by Battistella (2015);

4. Develop a USAR-based simulated environment on the Robot Operating System (ROS) to validate the coordination between deliberative-reactive robots and their behavior in face of unexpected situations. Providing a framework for the integration of the proposed and existing approaches.

1.2 CONTRIBUTIONS

Since this work approaches a system composed by multiple robots, a new centralized layer that act as go-between the human requiring tasks and the available robots has been introduced. To ensure the reliability and safety of the robots actions, the architecture proposed by Battistella (2015) has been implemented on each robot, but with the addition of a component responsible for updating the central system with the robot current status. Thus, in addition to the robot reactivity to the environment, the centralized task allocation layer also reacts to changes in the state of robots by reallocating the tasks if needed.

Therefore, this work addresses the following features through the implementation of an architecture composed of a centralized task allocation system and multiple robots embedded with a SCT-based control system:

1. As far as we know, we present the first set of formal models based on the SCT applied to the domain of USAR robots, resulting in a benchmark for future applications;

2. We integrate the deliberative-reactive architecture proposed by Battistella (2015) to an application of multiple heterogeneous robots, applied to a domain as relevant as USAR;

3. Achievement of a system that ensures reliability and safety for robots implemented in USAR operations, giving a step forward in the actual implementation of robots as an auxiliary tool in search of victims;

4. Deployment of a transparent system that allows the verification and reproduction of robots behaviors, ensuring the accountability of designers and operators due to the storage of the executed events and assigned tasks in a log file.

5. Development of an open-source framework implemented in ROS, which makes possible the simulation of the most diverse control systems integrated into the SCT-based architecture presented here.

1.3 DOCUMENT STRUCTURE

This document is organized as follows: in Chapter 2 we introduce some USAR definitions, and in the sequence the state of the art of robots applied to USAR and robotic systems based on formal methods are presented in Chapter 3. Following the initial review, we summarize the Supervisory Control Theory in Chapter 4. The following chapters present the main contributions of this work, starting with the development of formal models implemented in the robots control system, explained in Chapter 5. In this Chapter we delimit the applied USAR scene and formalize the problem of coordinating multiple robots in the SCT domain, modeling the system plants and specifications so that they ensure reliability and safety for the behaviors of robots. With the problem statement and system modeled, in Chapter 6 we explain the complete proposed architecture. In the sequence, Chapter 7 presents the evaluation of the proposed system in a simulated environment and the highlighted results. Finally, in Chapter 8 the concluding remarks and future perspectives are shown.

2 URBAN SEARCH AND RESCUE (USAR)

According to Murphy et al. (2008), **Search** is defined as the process of identification and location of survivors in a post-disaster situation, and **Rescue** is related to the release and removal of victims from the rubble. As a more complete definition, **Search and Rescue** represents the whole process of locating a victim, medical evaluation, appropriate removal, and medical treatment. **Urban Search and Rescue (USAR)** focuses on disasters involving collapsed structures in densely populated regions (MURPHY et al., 2008). Such situations may occur due to natural disasters (hurricanes, earthquakes, landslides) or caused by human action (structural failures, terrorism).

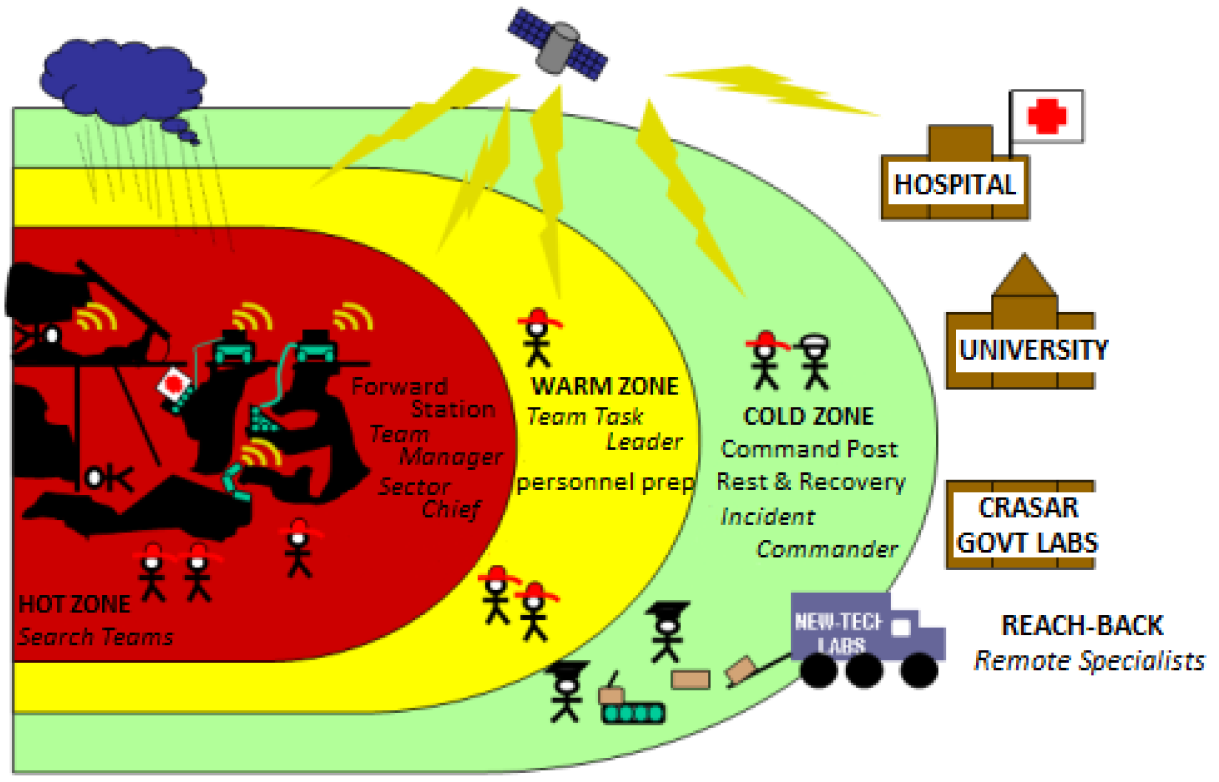
As USAR involves humans facing high risks, a number of protocols have been developed to maintain their physical and mental integrity. The most basic, but very important step, is the sectorisation and area assignment for the rescue workers. According Murphy (2004a) the teams are distributed in three distinct zones along the disaster area, as shown in Figure 1. The *Hot Zone* represents the most devastated area, where search and rescue efforts are concentrated. *Warm Zone* surrounds the most dangerous area and is where the workers prepare to enter the Hot Zone. The *Cold Zone* is the most external region of the restricted area and is where the Base of Operations is located, equipment is maintained, teams rest and communication with the population occurs.

The whole disaster operations are commanded by an Incident Commander (IC) responsible for prioritizing the sequence of missions to be accomplished by the teams (MURPHY, 2004a). According to the size of the Hot Zone, the area is separated into worksites assigned to Task Force Leaders who respond to the IC demands by planning strategies and commanding the functional teams. As can be seen in Figure 2, each Leader is responsible for supervising teams executing: search, rescue, and medical treatment of victims; hazardous materials removal; logistics of equipment and materials; and planning (MURPHY, 2004a).

The search for victims is very time consuming and adds high pressure to workers as the mortality rate exponentially increases and survivors extricated 48 hours after the event are unlikely to survive more than a few weeks (MURPHY, 2004a). Some techniques have been developed to assist workers performing such hazardous and time-sensitive tasks. Statheropoulos et al. (2015) highlights the following techniques as the most popular: physical void searching, audible call-out, search cameras, thermal images, electrical listening devices, and canine searching. Recently, robots have been occasionally applied and could fit into the command hierarchy according to the scheme shown in Figure 2.

Although great advances have been made in the past few decades, the relative number of rescued survivors remains low in relation to the total number of victims. In

Figure 1 – Disaster area classification. The most dangerous regions is the Hot Zone, where the rescue actions occur.



Source: (MURPHY, 2004a)

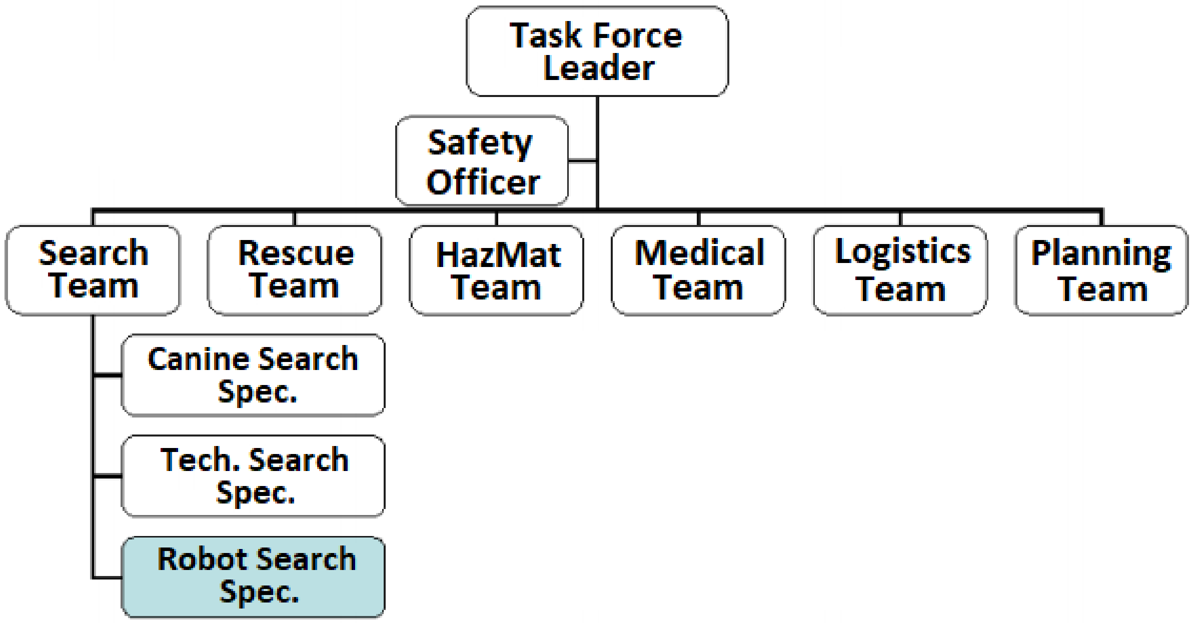
addition, the number of disasters involving collapsed structures has increased worldwide (STATHEROPOULOS et al., 2015). To improve the efficiency of rescues, protocols have been established by national organizations, such as the USA Federal Emergency Management Agency (FEMA), as well as by international groups, such as the International Search and Rescue Advisory Group (INSARAG). Below we summarize the protocols proposed by the INSARAG, which have been broadly implemented not just by international rescue teams, but also by national teams working at local events.

2.1 INSARAG PROTOCOLS

Due to the difficulties in coordinating rescue teams in major catastrophe events, the International Search and Rescue Advisory Group was founded in 1991 with the objective of establishing rescue standards to be followed by international teams involved in global disasters. Since then, the INSARAG protocols have been considered as the basis for search and rescue activities at different levels of disaster. Those which have a low, medium, or high need for specialized teams

These protocols are published in three volumes, which are available at INSARAG

Figure 2 – USAR team command hierarchy. Robots have been mainly applied to search activities.



Source: (MURPHY, 2004a)

(2015) and provide standard methodologies for countries and international USAR teams responding to large-scale structural collapse disasters. To improve the tasks efficiency and resources allocation, the planning process is summarized in Manual B from INSARAG (2015) Volume II by the following steps:

1. Assess the disaster current status;
2. Establish/obtain the objectives of the incident;
3. Develop a plan of action and disseminate it to the team;
4. Request the necessary resources to perform the tasks;
5. Execute the plan, monitor its progress and replan if necessary.

According to Manual B from INSARAG (2015) Volume II, the USAR operations should follow the following five operational levels, summarized in Table 1.

1. Wide Area Assessment;
2. Sector Assessment;
3. Rapid Search and Rescue;
4. Full Search and Rescue;

5. Total Coverage Search and Recovery.

Table 1 – USAR Operational Levels and their respective objectives.

Level	Definitions and Objectives
1	Preliminary survey and fast visual check of the affected area; Determines incident scope and magnitude and resource needs; Accomplish the sectorisation and establish priorities.
2	Sectors fast but methodical assessment to identify viable live rescue sites; Aims to assess the whole sector in a timely manner to make a plan of action; Has as result worksites assigned to USAR teams.
3	All allocated structures must be searched fairly rapid, maximising lifesaving opportunities; Modest commitment to each site with use of physical, canine and technical search tools; Limited penetration into the structures/rubble; Deeply entombed victims may not be found at his level, but structures where their search is worthwhile should be identified by teams.
4	Should identify, locate and rescue heavily trapped and entombed survivors; Teams penetrates into most or all survivable voids; Longer term operations that require wide range of USAR skills and involve several teams.
5	At this level, the chances of finding survivors are rare; The objective is to recover deceased victims; Characterized by the use of heavy machinery to clear rubble piles; Generally not carried out by international USAR teams.

Source: INSARAG (2015) Volume II

The INSARAG (2015) Volume III procedures guide presents some field and tactics information relevant to mission planning. Below we present some operational procedures that are applied to search and rescue teams.

1. Search:
    - Determine search and reconnaissance strategy;
    - Apply acoustic/visual equipment or assign canine teams when needed;
    - Ask for additional information from locals and first responders;
    - Ask opinion of specialist about structures reliability;
    - Constantly try to make contact with victims;
    - Collaborate with rescue and medical teams;
    - Ensure rest time for human and canine teams;
    - Stay updated about found victims and danger conditions.
2. Rescue:
    - Labor safety and security plans and brief the team;

- Define strategy and required equipment for victims removal, based on existing information;
- Make contact with medical team to ensure fast and safe medical care of victims;
- Ensure the safety of the working place along procedures;
- Always have an evacuation point defined and make use of individual protection equipment.

2.2 USAR LIMITATIONS

To ensure the best performance of the USAR teams, the procedures are constantly being reestablished. Statheropoulos et al. (2015) attempt to discover what delays USAR operations by a study on performance of teams in Port-au-Prince (Haiti,2010) and L'Aquila (Italy, 2009) natural disasters. The paper highlights the procedures applied; identifies points that might reduce the performance of teams, see Table 2; and proposes the improvements presented in Table 3 as a solution for increasing the efficiency of the activities executed by the teams.

Table 2 – Issues that prolong USAR operations.

• Rescue teams mobilization and dispatch to disaster, especially from abroad;
• Authorizations and permissions;
• Risk analysis and safety assessment;
• Planning and teams tasks assignment;
• Building re-searching after debris removal;
• Structural analysis;
• Empirical-based triage for prioritizing searching points is not sufficient in many cases;
• Limited amount of human resource and equipment. Canines are the most common searching tool.

Source: (STATHEROPOULOS et al., 2015).

As we can see from the Statheropoulos et al. (2015) analyses, improvements in environmental information acquisition and current state awareness are fundamental to increase the efficiency of rescues, as is the need for safe USAR techniques. These points would make great use of the implementation of robots, given the possibility of instrumentation with different sensors and access to conditions that are dangerous to humans. In addition, with the development of coordination techniques, multiple robots could more quickly scan the entire environment, leaving it to humans to carry out the rescue of victims, which requires a specific approach for each case faced.

Table 3 – Points that should be prioritized to increase performance.

• Correct decision-making based on initial info about the disaster (clear picture of what has happened, scale of disaster, available resources);
• Early detection and location;
• Safety of rescuers;
• Better logistic and available equipment;
• A more precise information exchange and planning;
• Use of easily deployed tools;
• International teams collaboration;
• Fast transportation of teams to the disaster site;
• Maintenance of a well understanding about the actual scenario.

Source: (STATHEROPOULOS et al., 2015).

2.3 USAR ROBOTS

The September 11, 2001 attacks to the *World Trade Center* are globally known for their proportions and devastation, and in the robotics field, it is known as one of the first implementations of robots in a USAR disaster. The use of robots was motivated by the need to access sub-human voids and it was accomplished by the collaboration between universities and governmental institutions like FEMA. In general, each robot was teleoperated by two humans, and the main objective was to search for victims in confined spaces inaccessible to humans and canine teams due to safety and space restrictions (MURPHY, 2004a, 2004b).

Since then, studies about the use of robots on USAR have been developed due to their capability of working in places where humans and canine teams do not fit, and because their use avoids putting humans in dangerous situations. According to Liu and Nejat (2013), rescue teams are subject to physical injuries, respiratory illnesses due to pollutants and gases, as well as the possibility of psychological trauma. By contrast, robots are immune to disease, do not suffer from fatigue and can work for several hours depending on battery availability; do not get stressed out; can be implemented in large quantities, improving the speed of rescues; and are disposable (LIU; NEJAT, 2013). Murphy (2004b), Williams, Sebastian, and Ben-Tzvi (2019) and Delmerico et al. (2019) highlight the following as the most common applications of robots in the USAR field:

- Victims search;
- Environment understanding for better planning;
- Search for path through rubble;

- Structural inspection;
- Identification of dangerous materials.

Robots have also been applied to tasks not directly related to the search procedures, as highlighted by Casper and Murphy (2003):

- Communication with victims;
- Provision of water and first aid
- Victims removal;
- Medical equipment transportation.

The robots at the WTC disaster were mainly used to search into inaccessible voids. They were inserted into the cavities, then the search for survivors was accomplished by the coordination between two members of the search team: one responsible for controlling the robot and another assigned to the monitoring of images acquired by the robot, who also helps the teleoperator by giving advice about the movements of the robot. Sketches of the environment were usually implemented as a tool for locating robots and determining their next movements.

After Murphy (2004b), **teleoperated** robots were applied to other USAR situations, making efficiency dependent on the performance of the operator controlling the robots. According to Liu and Nejat (2013), human operators are subject to stress, disorientation, and physical-cognitive fatigue. Such conditions reduce their alert and concentration levels, deteriorating their ability to identify victims and make decisions.

To reduce dependency on humans and improve efficiency, recent studies have approached robots with a higher autonomy degree, making them capable of decision making without human intervention. Doroodgar, Liu, and Nejat (2014) introduce a deliberative methodology to the robot, making it capable of deciding the most appropriate way to carry out the search for victims.

With the emergence of the capability of robots to plan actions, which is the most appropriate autonomy degree for robots applied to USAR? To answer this question, Liu and Nejat (2013) and Doroodgar, Liu, and Nejat (2014) present comparative tests in relation to the autonomy given to robots applied to search and rescue. Tests were carried out in simulated USAR environments, comparing robots with three control autonomy degrees: teleoperated; semi-autonomous, and fully-autonomous. Considering the obtained results, there is still a need for improvements to make them fully-autonomous and applicable to real situations. Thus, the best approach for actual technologies would be the semi-autonomous, with the collaboration between humans and robots. By this approach, low-level tasks (e.g., path planning) are responsibility of robots, and higher-level planning is now carried out by an operator (e.g., decisions about which tasks must be performed). Under these conditions, a single operator can be responsible for a team of robots, defining their tasks or taking control in situations where the robot is not able to act.

Although there are already a considerable number of studies about the implementation of multi-robots in USAR, they are not widely used in real situations. According to Delmerico et al. (2019), the implementation of robots in the field depends on:

- Simplicity of use: robots should be easy to understand, so that rescuers can be quickly trained in using all the tools provided by the multiple robot system;
- Commercially available: robots in action can suffer several damages and, therefore, it is necessary to have easy access to replacement parts;
- Robustness and reliability: because the environment is too delicate, involving the lives of people, such systems must guarantee minimum safety conditions for their users and for victims being rescued. Studies with Artificial Intelligence techniques are already well advanced in robot control, but these do not guarantee the necessary reliability for its implementation in real situations.

2.4 CHAPTER CONCLUSIONS

In this chapter we have summarized the international protocols approached by search and rescue teams. It was possible to notice that the procedures flow from a broader analysis of the whole disaster to the search of smaller areas with high chances of finding victims. Thus, three main steps can represent the baseline procedure: 1. reconnaissance of the site; 2. surface search; 3. voids search. Considering this idea of progressive search, in Chapter 5 we propose for the robots a set of maneuvers that would fit into these procedures. The protocols also highlight the need to maintain the safety of teams in first place, as well as constant situational awareness and dynamic planning. The models proposed in Chapter 5 follow these assumptions, and the possibility of constant replanning was considered in the architecture explained in Chapter 6.

It was also important to review the usual limitations faced by the search and rescue teams to help us delimit relevant points to be approached by the proposed method. The need for better understanding of the disaster area, maintenance of the safety of teams and faster deployment were considered in the specifications applied to the synthesis of the controllers, presented in Chapter 5.

We have also observed that robots applied to the USAR domain have shown to be very beneficial for speed improvements and teams safety increase. Robots can perform faster sectors scans, have the potential to penetrate regions inaccessible by humans or dogs, search for survivors and hazardous materials, and even verify structural conditions. However, despite the last two decades advances in USAR robotics, there is still much room for improvements. Works have mainly developed techniques to improve the performance of robots, but the robustness, reliability, and safety of how they execute their actions still require more efforts. This need motivated the development of the SCT-based system and is the baseline for the definitions of the specifications models presented in Chapter 5.

3 SYSTEMATIC REVIEW

In this chapter, we summarize the specialized technical literature, dividing it into two main parts: initially, we analyze works that address the more general use of robots in search and rescue tasks; then, those approaches that make use of Discrete Event Systems techniques to synthesize multi-robots systems are presented, regardless of the application.

3.1 ROBOTS IN USAR DOMAIN

3.1.1 Method for articles search

In order to identify works that implement robots in USAR, we searched four international specialized journal databases (see Table 4 columns) for articles published after the year 2000 that included the keywords listed in the rows of Table 4. Among a total of 254 works, we selected – based on an analysis of the abstract content – those that did not deal exclusively with constructive aspects of the robot or with low-level control strategies, such as trajectory planning. After this step, 50 articles remained. However, of these 50, only 25 presented strategies for the coordination and control problems at a level of detail adequate for understanding and evaluating their advantages and disadvantages. Of these 25, we will not address below those dealing with the development of interfaces for teleoperated systems as a form of human-robot interaction and improvement of situational awareness. Therefore, the next section deals exclusively with the content of the eight articles most relevant to the topic of this research.

Table 4 – Systematic review of implementation of robots in USAR.

Search term	**Source**				
	IEEE Xplore	Springer Link	ACM	Journal of Field Robotics	Total
(“urban search and rescue” OR “USAR”) AND (“robots” OR “robotics”)	23	173	16	42	254
+ “coordination”	3	73	1	27	104
+ “robustness”	4	57	5	27	93
+ (“human robot interaction” OR “HRI”)	5	53	11	0	69

3.1.2 Review of Selected Articles

Doroodgar, Liu, and Nejat (2014) and Liu and Nejat (2016) seek to reduce the USAR robots dependency on humans teleoperation, which generally leads teams to stress and cognitive and physical fatigue. To increase the degree of autonomy, Reinforcement Learning (RL) techniques are implemented to improve the task execution of a semi-autonomous robot working in unknown environments. The approach reduced the exploration time and chances of collision compared to fully teleoperated approaches. The RL can make the robot adaptable to unexpected situations that are very common in USAR. But as it is based on training on a broad set of simulated conditions, the behavior of robots becomes very dependent on the specific set of situations addressed by the training stage.

A big issue in large urban disasters is the lack of situational awareness immediately following the catastrophe. Intending to quickly map the environment and point out superficial locations of victims, Arnold, Yamaguchi, and Tanaka (2018) propose the use of Behavior-based Artificial Intelligence as a framework for controlling a swarm of Unmanned Aerial Vehicles (UAV) exploring an unknown USAR environment. Each robot defines the most suitable behavior based on a hierarchy table and its knowledge of the environment. Swarms are capable of accomplishing collective behaviors only by combining individual behaviors, with no centralized control system. In simulations with up to 20 UAVs, the approach resulted in great improvements in the SA acquisition, accomplishing more than 90% coverage of an area of 2 km^2 in less than 2 hours, something that would require even days by traditional techniques.

Although these AI-based approaches show great results, they have the inconvenience that there is no mathematical guarantee that the robots will accomplish the desired behavior. In this work, we implement the semi-autonomous concept as done by Doroodgar, Liu, and Nejat (2014) and Liu and Nejat (2016) and determine robots actions based on a priority table, as approached by Arnold, Yamaguchi, and Tanaka (2018). But here the basis of the whole system is based on the SCT, which has the advantage of being a mathematically proved method. As a result, we guarantee that the projected constraints will always be satisfied.

Considering this need for modeled reactivity of the robot in the face of unexpected events, Carbone et al. (2008) introduce a mixed proactive and reactive approach to control the exploration of an autonomous robot in unstructured and initially unknown USAR environments. By the proposed method, the robots decision-making is restricted by a high-level executive layer containing causal and temporal relations expressed as declarative models by Golog language. This layer determines the behavior allowed in each condition, e.g., “if a victim is found, the robot cannot continue exploring the environment”.

In another approach, Talamadupula et al. (2010) addresses task planning for an autonomous robot that cooperate with humans in open environments (*Open Worlds*), i.e., where the rescue team must perform a task without prior knowledge of the environment. In this context, the tasks and goals associated with them depend on uncertain information, e.g., the objective “to report the existence of a victim” depends on knowledge that the rescue team does not have, the location (or even the existence) of the victim. To solve the problem, the authors propose a strategy capable of dealing with *conditional objectives*, which are transformed into plans only in specific situations. Both Carbone et al. (2008) and Talamadupula et al. (2010) address the same problem as this work: increasing the reliability and safety of the behaviors of robots. These have significantly increased the robustness of the robot's behavior in unknown environments. However, they do not address the robustness of robots to the occurrence of failures and are not applied to multiple robots.

Post-disaster environments present very unreliable communication due to the interference caused by debris composed of metallic materials. To overcome this problem, Vittorio Amos Ziparo et al. (2007) and Kleiner and Dornhege (2007) present a possible solution for the coordination between robots even in the presence of communication failures.

In the article, the use of RFID tags (*Radio Frequency Identification*) is proposed as a means of indirect communication between robots. Robots release RFID tags at newly explored points and insert information from their current map into them. When passing over tags already present in the environment, the robots read the stored content, referring to robots that have already passed through there. With this information, it is possible to coordinate the exploration, forwarding the robot to unexplored regions and merging the maps of all robots that passed through the same point. The strategy developed by the authors was used to coordinate 12 robots in a mapping task, with satisfactory results. The work above is one of the few that considers the difficulty of communication, which is usual in USAR environments. In this work, we do not ensure the coordination between robots in case of communication failures, as Vittorio Amos Ziparo et al. (2007) and Kleiner and Dornhege (2007) did, though the control system of robots, based on formal models, ensure that the robots behave safely in such conditions.

The problem of removing road debris using autonomous and cooperative robots is discussed in Nath, Arun, and Niyogi (2019). In the article, the main goal is to coordinate the robots so that, together, they remove obstacles that are impossible to be removed by only one robot. For this, the authors modeled the communication between the robots by means of finite automata. The strategy was tested in a simulated environment, showing that it is capable of coordinating up to 10 robots to perform coordinated activities. Furthermore, the method developed by the authors proved to be robust in the case of robot failures, given that the missing robot can be easily replaced by another one without the need for *offline* re-planning. The work, however, did not address the interface between the robots and a team of human beings.

3.2 MULTI-ROBOTS COORDINATION IN DISCRETE EVENTS SYSTEMS LITERATURE

3.2.1 Method for articles search

At this stage, we searched the literature for articles published after year 2000 focusing on multi-robots that, regardless of the application (see Table 5): a) adopt Petri Nets as a formalism for modeling and analyzing the behavior of the system; and/or b) synthesize supervisors based on the Supervisory Control Theory (SCT) of Ramadge and Wonham (1987).

Table 5 – Systematic review of formal methods applied to multi-robots systems.

Search term	**Source**				
	IEEE Xplore	Springer Link	ACM	Journal of Field Robotics	Total
(“multirobot” OR “multi-robot”)	289	1970	53	336	2648
+ “petri net”	5	47	1	2	55
+ “supervisory control theory”	0	6	0	17	23

Although the searches carried out resulted in 78 articles, several of them were related to supervisory systems and not to the Supervisory Control Theory according to the Ramadge and Wonham (1987) approach. As for applications with Petri Nets, only works focusing on the use of such a formalism to obtain a model to be used in analysis (e.g., existence of blocks) were considered, and not merely as a graphical representation of the system’s behavior. In total, nine articles were selected for in-depth reading.

In addition to the articles selected according to the method above, we will present the works developed by members of this research group. Battistella (2015) and Carolina de Lima (2019) implemented the SCT to control a submarine robot and multi-robot patrolling, respectively. These works employ a multi-layer control architecture, on which we will base the development of this project.

3.2.2 Review of Selected Articles

Petri Nets have received a lot of attention from works because they are a formal methodology that is easy to interpret due to their compact and easy-to-understand graphical representation. It allows modeling, analysis, and simulation of distributed and concurrent discrete event systems (EBADI; PURVIS, Maryam; PURVIS, Martin, 2010). Ebadi, Maryam Purvis, and Martin Purvis (2010) and Farinelli et al. (2017) model their robotic systems using Coloured Petri Nets (CPN). The first work models the allocation of collaborative tasks in a multi-robots system involving up to 100 robots. The models were proved right by being implemented on four agent selection strategies for which the models demonstrated adequacy. As for the second work, this one used CPNs to model interruptions performed by an operator without the need for a complete system shutdown, e.g., removing a robot from the team without having to stop the others.

Before applying robots to dynamic, unstructured, and partially unknown environments, it is fundamental to correctly design the possible behaviors executed by them. Vittorio A Ziparo et al. (2011) presented a systematic approach called *Petri Net Plans* (PNPs), which allows the development of distributed models approaching collaboration and cooperation relationships between robots performing activities according to their knowledge of the environment. Also, Costelha and Pedro Lima (2012) propose a framework for modular modeling in Petri Nets based on three layers: the action coordination layer; the action execution layer; and the layer of the environment. These templates can be created separately and subsequently composed, which facilitates the design of the models and allows the task to be analyzed as a single integrated model. Such techniques based on Petri Nets have proved to be robust because they use formal models that can be verified, but they do not guarantee their correct adequacy to the real behavior of the system, as they do not follow a formal controller synthesis process, as obtained by the Supervisory Control Theory.

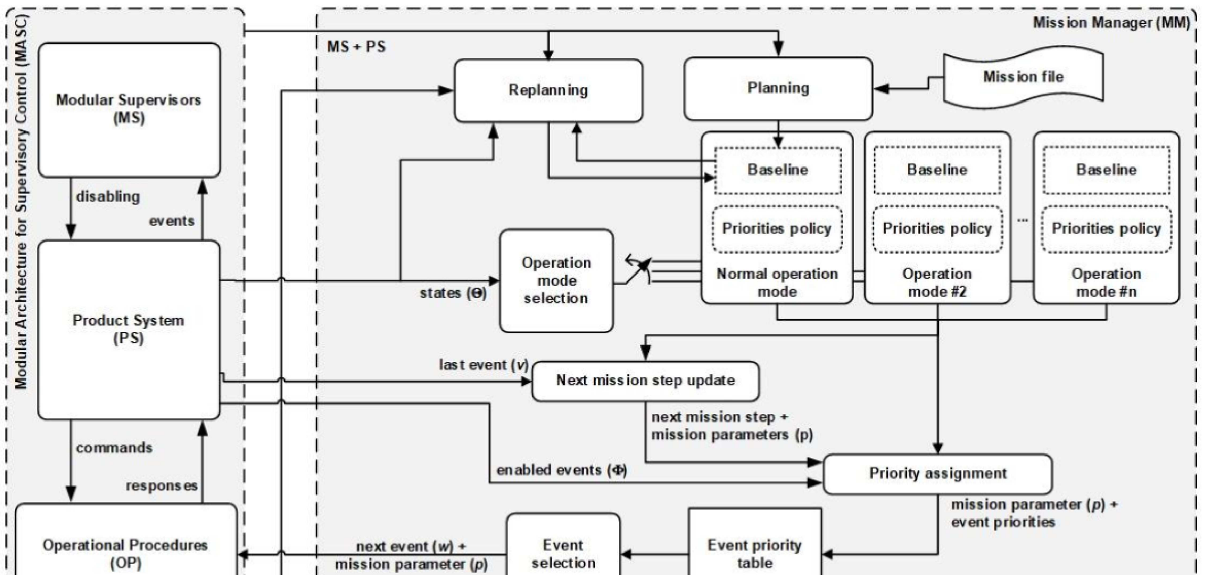
Formal models can also help in determining possible paths for the robots. Koo et al. (2012) and Kloetzer and Mahulea (2019) aimed to develop path planning systems involving multiple robots acting in a coordinated manner. To do so, they modeled the environment using formal models represented by automata and Petri Nets, respectively. Based on formal specifications of the objective to be performed, the proposed models are checked, and this verification results in the most suitable trajectory for the robot. The approaches result in systems that guarantee safe plans, but they become highly restrictive as they result in very limited sets of allowed actions. In USAR environments, the safest plan may not be the most efficient, so the robot must prioritize the safest one. The presented approaches can ensure the right plan selection, but can sometimes be very restrictive, resulting in only one possible plan or none at all. The SCT could be more feasible for planning the behavior of robots as it would generate minimally restrictive systems, unlike the approaches above.

Dai, Jiang, and Li (2019) present another implementation of formal models as a planning tool. But this time, the goal is to reduce the computational cost of exploring unknown environments and to minimize the scanning time when using multiple robots. A system that integrates predicate reasoning and reactive behavior based on a two-level architecture was proposed. The low-level makes use of predicates to determine the search behavior in a *greedy* way, while the high-level has a supervisor obtained by SCT in order to avoid collisions between robots. This integrated approach resulted in faster exploration and reduced downtime compared to the following techniques: *Repeated Auction Algorithm (RAA)*, *Decision-Theoretic Approach (DTA)* and *Supervisory based Control Coordination (SCC)*.

Besides avoiding unwanted situations, e.g., collisions, the SCT allows to define alternative behaviors for the system. Dai, Jiang, and Zhao (2016) sought to improve the efficiency of environmental exploration by coordinating tasks assigned to multiple robots. As communication difficulties can affect coordination, it is the main point addressed by the work, in which the Supervisory Control Theory was applied to control the behavior of robots according to the communication status. The robots were capable of executing different exploration approaches, and the best for the moment was selected by the supervisors synthesised by the SCT. Compared to two other auction-based methodologies, the proposal showed greater speed of exploration of the environment.

Battistella and Queiroz (2014) and Battistella (2015) aimed to control the missions of an Autonomous Underwater Vehicle (AUV) in partially unknown and dynamic environments, e.g., in hydroelectric lakes. A hybrid deliberative-reactive mission control architecture was proposed based on the Local Modular Supervisory Control Theory (LMSCT) proposed by Queiroz and Cury (2000), generating a minimally restrictive supervisor. The proposed architecture is shown in Figure 3, where the two main components interactions are described. To the left, the Modular Architecture for Supervisory Control (MASC) represents the implementation of the synthesised local supervisors that ensure the correct behavior of the robot. The Mission Manager (MM) represents the deliberative layer, being responsible for planning the best sequence of actions according to the controllable events enabled by the MASC. This work gives the required

Figure 3 – General structure of Mission Manager and its interaction with the MASC.



Source: (BATTISTELLA, 2015)

reliability to the robots and also makes possible their operation in degraded modes.

Carolina de Lima (2019) also makes use of this architecture, but with adaptations for the application of multi-robots acting in patrolling an environment, demonstrating that LMSCT is not limited to applications of only one robot. Simulations confirmed the ability of the system to safely react to unforeseen events based on events allowed by the synthesized supervisors.

As a last example of a Supervisory Control Theory application, Lopes et al. (2016) proposes its use in the synthesis of robotic swarm controllers. The work introduces SCT concepts regarding the synthesis of supervisors to control four behaviors: segregation, aggregation, objects grouping, and groups formation. It also applies the LMSCT and, through physical experiments, it is proven that the behavior is maintained according to conditions established by formal models.

3.3 CHAPTER CONCLUSIONS

The main features of the articles reviewed in this chapter are summarized in Table 6. We can observe that although none of them addresses all the following topics, they complement each other. Thus, the reviewed articles might be a good reference for the development of the proposed Master Thesis.

Table 6 – Scope of revised articles.

Articles	**Features**							
	A	B	C	D	E	F	G	H
(DOROODGAR; LIU; NEJAT, 2014)	x					x		S
(LIU; NEJAT, 2016)	x	x	x			x		S
(ARNOLD; YAMAGUCHI; TANAKA, 2018)	x	x	x					A
(CARBONE et al., 2008)	x			x				A
(TALAMADUPULA et al., 2010)	x					x		A
(ZIPARO, Vittorio Amos et al., 2007)	x	x			x			A
(KLEINER; DORNHEGE, 2007)	x	x			x			A
(NATH; ARUN; NIYOGI, 2019)	x	x	x		x			A
(EBADI; PURVIS, Maryam; PURVIS, Martin, 2010)		x						A
(FARINELLI et al., 2017)		x			x	x		A
(ZIPARO, Vittorio A et al., 2011)		x						A
(COSTELHA; LIMA, P., 2012)		x						A
(KOO et al., 2012)		x						A
(KLOETZER; MAHULEA, 2019)		x						A
(DAI; JIANG; LI, 2019)		x					x	A
(DAI; JIANG; ZHAO, 2016)		x		x			x	A
(BATTISTELLA, 2015)				x	x		x	A
(LIMA, C. de, 2019)		x		x	x		x	A
(LOPES et al., 2016)		x		x			x	A

Features: **A** - USAR; **B** - Multiple robots; **C** - Heterogeneity; **D** - Controller formally synthesised; **E** - Tolerant to communication or mechanical failures; **F** - Human-robot interaction; **G** - Minimally restrictive controller; **H** - Autonomy level: teleoperated (T), semi-autonomous (S) or fully-autonomous (A).

4 SUPERVISORY CONTROL THEORY REVIEW

In this chapter we introduce relevant concepts about the Discrete Event Systems and the Supervisory Control Theory with the intention of giving the reader the needed knowledge to understand the implemented technique. For a more detailed explanation, we suggest the book from (CASSANDRAS; LAFORTUNE, 2009).

4.1 DISCRETE EVENT SYSTEMS (DES)

Discrete Event Systems are characterized by a discrete set of states and a set of possible events Σ (alphabet) that influence the system dynamic (i.e., change of states) (CASSANDRAS; LAFORTUNE, 2009). For example, a state can represent that “a robot is currently executing a maneuver”, and an event could signal that “a victim has been found”.

A possible representation of the behavior of a DES is through a language. The Theory of Formal Languages and Automata, better explained by Cassandras and Lafortune (2009), defines a language as a subset $L \subseteq \Sigma^*$, with Σ^* representing the set of all finite strings composed by elements from Σ. In other words, the language L is the set of all possible event sequences generated by the system.

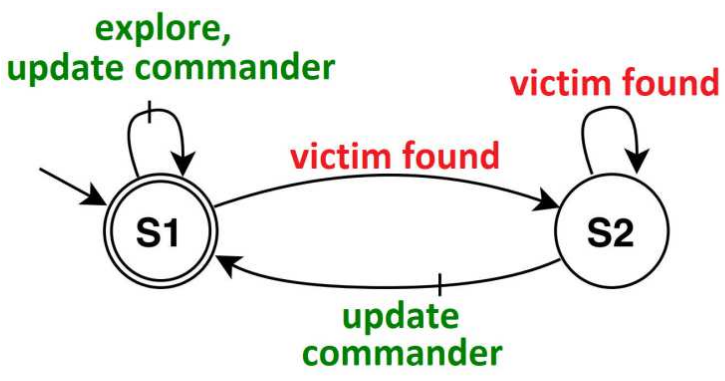
A language can also be represented by a graphical representations. An automaton, according to Ramadge and Wonham (1988) and Cassandras and Lafortune (2009), is a quintuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, where:

- Q - finite non-empty set of states;
- Σ - a finite set of events (alphabet);
- $\delta : \Sigma \text{ x } Q \rightarrow Q$ - states transition function;
- $q_0 \in Q$ - initial state;
- $Q_m \subseteq Q$ - set of final states.

This representation of languages by automata provides a more visual way of representing the behavior of a language. An automaton is a state transition diagram represented by directed graphs like the one presented in Figure 4. The initial state is highlighted by an arrow, marked states (that represent the goal) are represented by double circles, and the transitions between states are defined by the arcs. This representation allows the modeling of discrete event systems as well as the implementation of supervisory and diagnostic systems responsible for controlling or monitoring the behavior of a specific system (CASSANDRAS; LAFORTUNE, 2009).

An automaton can represent two possible languages: the generated language $L(G)$ which represent every possible events sequence, and the marked language $L_m(G)$

Figure 4 – Automaton example. The floating arrow points to the initial state, double circles represent the goals and arcs correspond to the events that determine the dynamic of the system. Slashed arcs mean that the event is controllable.



composed by all strings that lead from an initial state to a marked one. Since the marked language is restricted by the marked states and the generated language follows the automaton dynamic, then $L_m(G) \subseteq L(G) \subseteq \Sigma^*$.

According to Ramadge and Wonham (1988) and Cassandras and Lafortune (2009), for a language $L \subseteq \Sigma^*$, we define as prefix-closure of L the set composed of all Σ^* sequences that are prefix of L:

$$\overline{L} = \{s : s \in \Sigma^*, st \in K, \text{ for some } t \in \Sigma\} \tag{1}$$

To ensure the liveness of the modeled system we must ensure that there is no deadlocks (non-marked states with no output event). It can be accomplished by satisfying the condition $\overline{L_m(G)} = L(G)$, that means that all strings in $L(G)$ have to represent a sequence of events that lead to a marked state or to a state from which at least one marked state is accessible.

Complex systems can generally be accomplished by the composition of smaller sub-systems. In such situations, the behavior of complex DES can be represented by the parallel composition between the automata that model the sub-systems (CASSANDRAS; LAFORTUNE, 2009). The parallel composition of two automata R and G is represented by $R||G$ and is defined as the cuncurrent execution of both R and G. Events that are part of both alphabets can be executed only if possible in both automata, and non-shared events can be executed if active in the automaton from where it belongs (CASSANDRAS; LAFORTUNE, 2009).

4.2 SUPERVISORY CONTROL THEORY (SCT)

The Supervisory Control Theory (SCT) proposed by Ramadge and Wonham (1987) defines a synthesis method for DES closed-loop controllers. The procedure is accomplished by processing the system behavior (plants) and desired constraints (specifications) modeled by means of formal automata. It results in a feedback control structure (Supervisor), if it exists, which, when interacting with the plant, ensures that the specifications will be met in a minimally restrictive manner. That is, the supervisor allows the largest set of **controllable** chains in the plant that guarantees the established specifications.

The automaton example from Figure 4 illustrates a specification that models a constraint to be satisfied by the closed-loop system. It states that initially, the robot can carry out the exploration of the environment or update the commander regarding the conditions of the environment until a victim is identified. After finding a victim, the robot must prioritize the updating the commander and may resume exploration only after informing the commander about the location of the victim. The transition represented by the green color (slashed arc) is *controllable* ($\Sigma_c \subseteq \Sigma$), making the robot responsible for their execution. Highlighted in red, the event “victim found” is determined by means of sensors, making it impossible for the system to define the exact moment of its occurrence, thereby characterising it as *non-controllable* ($\Sigma_u = \Sigma - \Sigma_c$). We can also notice that the state **S1** is **marked**, representing the interest in remaining in this state, suggesting that the exploration of the environment should be carried out whenever possible.

To ensure that the system executes only sequences of events that satisfy the specifications, Ramadge and Wonham (1988) theory proposes the representation of a control mechanism Γ that enables and disables the possibility of an event occurrence. The system defines an event as enabled if $\gamma(\sigma) = 1$ and disabled if $\gamma(\sigma) = 0$. Thus, as it is impossible to control events contained in Σ_u, the set of control parameters of a system G is given by:

$$\Gamma = \{\gamma : \Sigma \rightarrow 0,1, \gamma(\sigma) = 1 \text{ for each } \sigma \in \Sigma_u\} \tag{2}$$

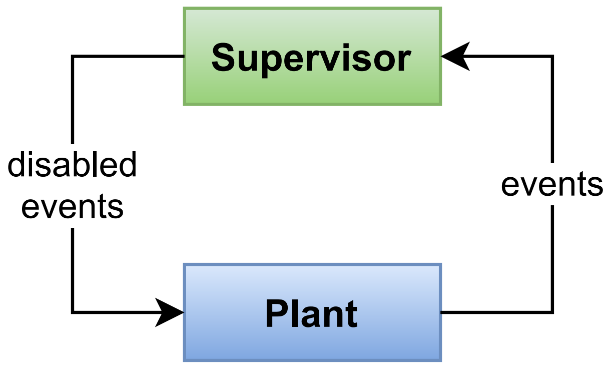
So it is possible to represent the process as $G_c = (\Gamma \text{ x } \Sigma, \Sigma, \delta_c, q_0, Q_m)$, with $\delta_c : \Gamma \text{ x } \Sigma \text{ x } Q \rightarrow Q$ represented by:

$$\delta_c = \begin{cases} \delta(\sigma, q) & \text{if } \gamma(\sigma) = 1 \\ \textit{undefined} & \text{otherwise} \end{cases} \tag{3}$$

Based on the presented definitions, it is possible to define a Supervisor S controlling the process G_c as an automaton $S = (\Sigma, X, \xi, x_0)$ together with a map of feedback $\varphi : X \rightarrow \Gamma$. As shown in Figure 5, the supervisor monitors the sequence of events occurred in plant G_c and, depending on its own current state, keep enabled only transitions that guarantee the system desired performance.

As some events are not controllable, the SCT becomes dependent on the controllability of the plant, which is based on the existence of controllable events amid chains

Figure 5 – Monolithic supervisor closed control loop.



Source: Adapted from Queiroz and Cury (2002a)

of actions that result in undesired situations. A specification K is considered controllable related to a plant G if $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$, that is, the occurrence of a non-controllable event in the plant G does not trigger a sequence of events that disagree with K. The controllability and non-blocking guarantee is the necessary and sufficient condition for the existence of a supervisor S that satisfies a specification K.

The synthesis of a supervisor S could be summarized by the following steps:

- Modeling the plant: representing the expected behavior of the system by means of formal models represented by automata, delimiting the controllable and non-controllable events;
- Modeling the specifications: defining the constraints of the system by blocking the occurrence of events to avoid undesired situations;
- Synthesising the controller: defining the most permissive supervisor, that is, ensuring that the necessary conditions are met with a model that maximizes closed loop language.

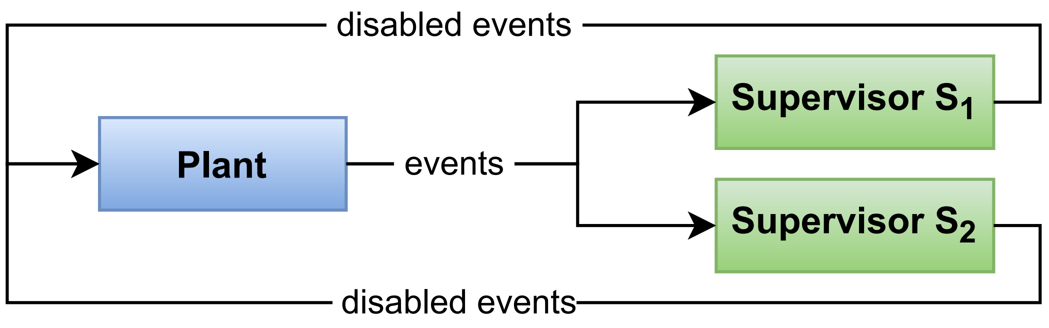
4.3 LOCAL MODULAR SUPERVISORY CONTROL THEORY (LMSCT)

Although the Supervisory Control Theory proposed by Ramadge and Wonham (1987) proves to be excellent for the synthesis of robust and minimally restrictive controllers, it has the state explosion as a major limitation. Models that are too complex can result in the presence of large numbers of states, making their real implementation unfeasible due to the amount of memory and processing required. When dealing with multi-robots systems, the state explosion problem limits the scalability of robots used by the system.

To overcome this issue, Ramadge and Wonham (1988) proposed the modularization of supervisors into a set of smaller and specialized supervisors, defined as Modular

Supervisory Control Theory (MSCT). The new approach synthesises supervisors that act in parallel in controlling which events are enabled to be executed by the plant, as can be seen in Figure 6.

Figure 6 – Modular Supervisors integration with the plant.

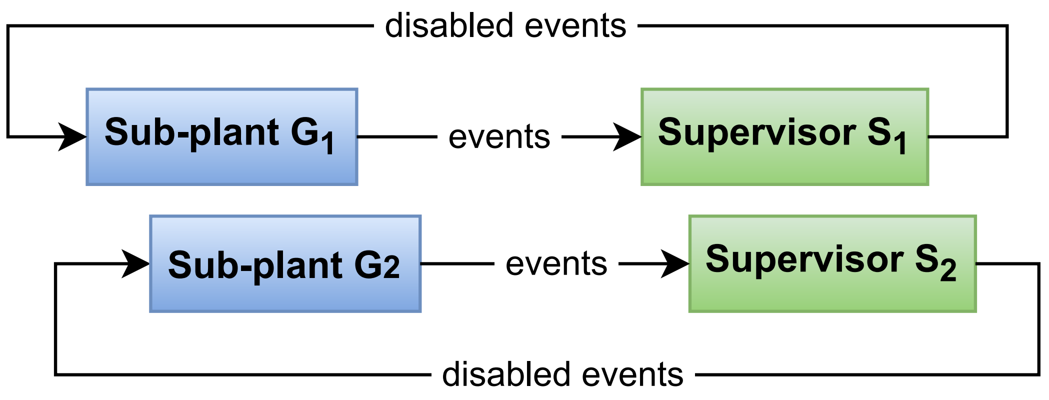


Source: Adapted from Queiroz and Cury (2002a)

The Local Modular Supervisory Control Theory (LMSCT) proposed by Queiroz and Cury (2000) extends the MSCT by making better use of the decentralized characteristics of large systems. In general, a system can be split into smaller specialized subsystems. When these subsystems are not all interrelated, the LMSCT can be applied to reduce the computational effort needed for the synthesis and implementation of supervisors.

By implementing smaller subsystems, it is possible to synthesize supervisors focused on each sub-plant, as shown in Figure 7. As a result, we end up having fewer supervisors with fewer states, making it more feasible for real applications.

Figure 7 – Local Modular Supervisor interaction with sub-plants.



Source: Adapted from Queiroz and Cury (2002a)

The LMSCT applies the concept of Product System Representation (PSR), so that the system subplants, represented by $G_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi})$, can be combined when synchronized with each other. The goal is to keep the largest possible set containing asynchronous and distinct subsystems from each other. Local plants $G_{loc,j}$ are defined for each specification $E_{gen,j}$ based on the parallel composition of the subsystems G_i that have events in common with $E_{gen,j}$.

Local specifications $R_{loc,j}$ are obtained from the parallel composition between $G_{loc,j}$ and E_j. Then, each modular local supervisor $S_{loc,j}$ is obtained by calculating the maximum sub-language of K that is controllable and non-blocking. Since the local supervisors control the system in parallel, it is needed to ensure that there is no conflict between each other. It is accomplished if the parallel composition of all $S_{loc,j}$ is non-blocking.

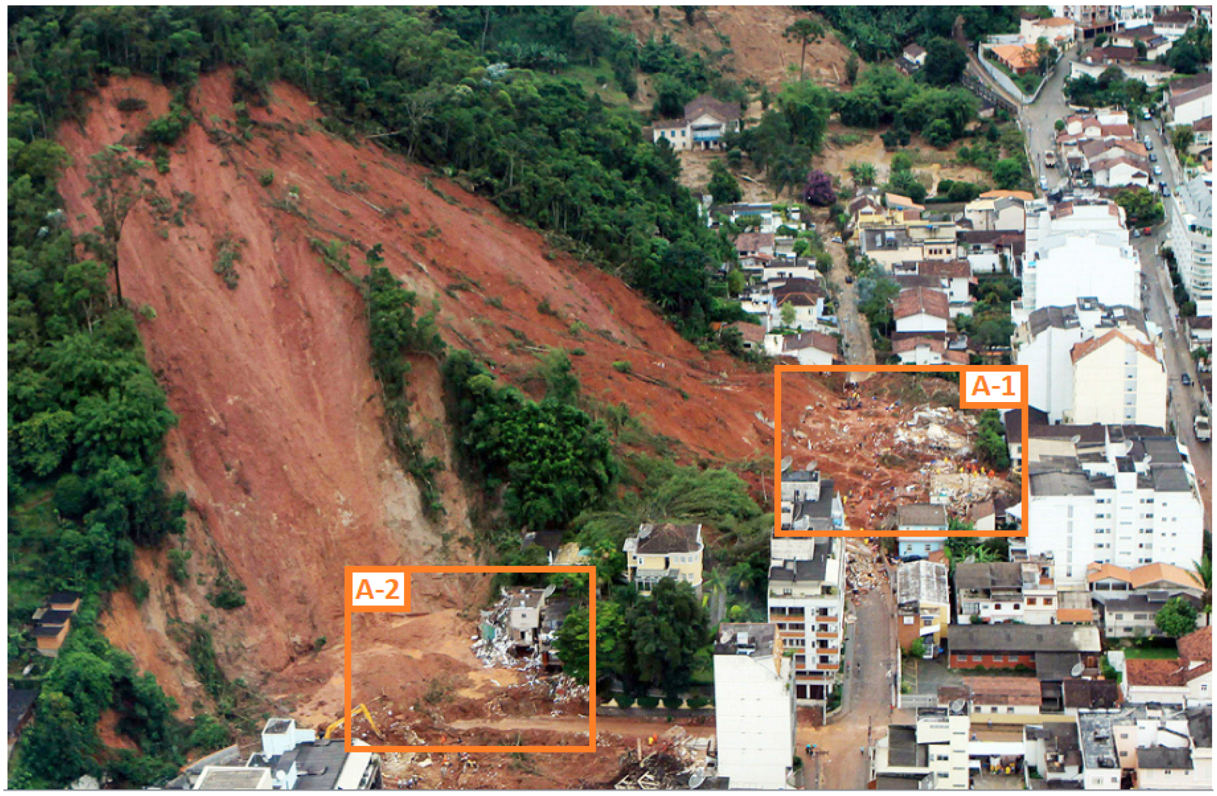
5 USAR MULTI-ROBOTS AS A SUPERVISORY CONTROL PROBLEM

In this chapter, we present a post-disaster scenario often faced by USAR teams, in which the use of mobile robots can be beneficial. Next, we delimit it as a Supervisory Control Theory problem. Then the plant and specification models are summarized, followed by a demonstration of the synthesised supervisors.

5.1 MOTIVATION SCENARIO

An urban disaster very common in Brazil is the occurrence of landslides, as the one shown in Figure 8. Such disasters can have their area subdivided into worksites where search and rescue activities assigned by Task Force Leaders take place. The scene from Figure 8, for example, could be subdivided into worksites A-1 and A-2.

Figure 8 – An example of a landslide scene and possible worksites division. A landslide occurred in Nova Friburgo - RJ in January 2011.



Source: (SILVA CARDOSO; VIEIRA, 2016).

Usually, these environments do not grant sufficient initial understanding of the environmental distribution and are vulnerable to unpredictable events. The following characteristics are considered in this work:

- Presence of semi-collapsed structures;

- Injured people randomly distributed;
- Presence of hazardous materials, e.g., gas leakage;
- Occurrence of unexpected events such as failures of the robots, low battery levels, and new findings (victims or dangers).

In this scenario, we have summarized the command hierarchy by the image of a Task Force Leader responsible for defining sequences of tasks that must be executed by both human teams and robots. And due to the benefits provided by heterogeneous systems, where each component can have their own specialized capabilities, the following robots and payloads have been applied:

- UAV: an aerial autonomous vehicle outfitted with a RGB camera, a Depth Camera, and a Sonar for Simultaneous Localization and Mapping (SLAM) of the environment. We also considered that the UAV is equipped with a thermal camera for victim localization;
- UGV: an unmanned ground vehicle, represented here by a Pioneer-3AT, that is also equipped with RGB and Depth Cameras for 3D mapping. A thermal camera is applied for the localization of victims. In addition, a gas sensor is used to detect gas leaks.

Considering the described sensors embedded in the robots, in this work we have considered UAVs and UGVs composed by the following subsystems: *battery consumption monitor*; *failures detection*; *victim recognition*; *gas sensor* (only on UGV); and *communication*. Based on the missions presented by Murphy (2014) as usually assigned to robots, we have proposed the set of maneuvers listed in Table 7.

Table 7 – Maneuvers executed by UGVs and UAVs proposed based on the tasks usually assigned to robots, with a focus on search procedures.

Maneuver	**UGV**	**UAV**
approach (APP)	yes	yes
assessment (ASSESS)	no	yes
exploration (EXP)	yes	no
return to base (RB)	yes	yes
surroundings verification (VSV)	yes	yes
victims search (V_SEARCH)	no	yes
safe land (LAND)	no	yes
teleoperation (TELE)	yes	yes

The **approach** maneuver is responsible for sending the robot to any point of interest; on **assessment** the UAV rapidly moves over the worksite to acquire relevant info; the **exploration** is only executed by the ground robot and represents the full exploration of a region with the objective of finding victims and gas leaks; **return to base** makes the robot automatically return to the base of operations and execute an autonomous landing; **surroundings verification** represents the action of looking around a victim to increase SA; **victims search** is a more focused search where an UAV moves closer to the ground with the goal of finding partially buried victims; **safe land** executes an emergency landing with extra procedures to avoid landing on possible victims; and **teleoperation** models the direct human control of the robot.

5.2 DELIMITATION OF THE PROBLEM

The scenario, robots capabilities, and payloads from Section 5.1 were proposed based on the overview presented in chapters 2 and 3, representing a reasonable first scenario to be formally modeled. Considering the described maneuvers and subsystems and the need for a system to coordinate the robots, in this work we face the problem of determining a minimally restrictive logic that ensures the following characteristics for the system:

1. All the components required for the correct robot operation interact with no conflicts between each other;
2. The robots commit to the maintenance of the situational awareness of humans about the disaster;
3. Robots have their applicability increased, meaning they are still useful when not fully operational, as long as the safety of everyone close to them is ensured;
4. The tasks required by humans are executed only with all their dependencies met;
5. Prioritization of missions;
6. Optimization of the tasks assigned to the robot.

As was shown in Chapter 4, the synthesis of controllers by the SCT requires the modeling of two main elements: the plant and the specifications. For the proposed scenario, the components described in Section 5.1 can be considered the plant itself, since they represent the physical behavior of the robot, though some abstractions might be needed to simplify the controller. To constrain the system in a way to attain the first four characteristics highlighted above, we have proposed four specification classes, which are described in Section 5.4.

The coordination of the robots, with prioritization of missions according to the desires human team, is approached here by a deliberative layer. However, in addition to not directly implementing the SCT, this layer still interacts with the supervisors and reacts to changes in them.

5.3 PLANT MODELING

The global plant that models physical system dynamics of each robot was modeled by a set of automata that together represent the full system of a robot. According to Battistella (2015), depending on the control domain, low-level systems do not necessarily need to be modeled, so they can be considered as event generators and service providers, e.g., if the goal is to control tasks execution, the changes in continuous systems (like motor electrical current) might be irrelevant to the models. So we chose to model for the robots the maneuvers described in Table 7 in a high-level abstraction of their actual behavior. Besides the maneuvers, the sensors, battery and failures monitors, and communication subsystems were also modeled. In the following subsections, we introduce the modeled automata that represent the dynamics of these subsystems.

5.3.1 Subsystem 1: General maneuvers

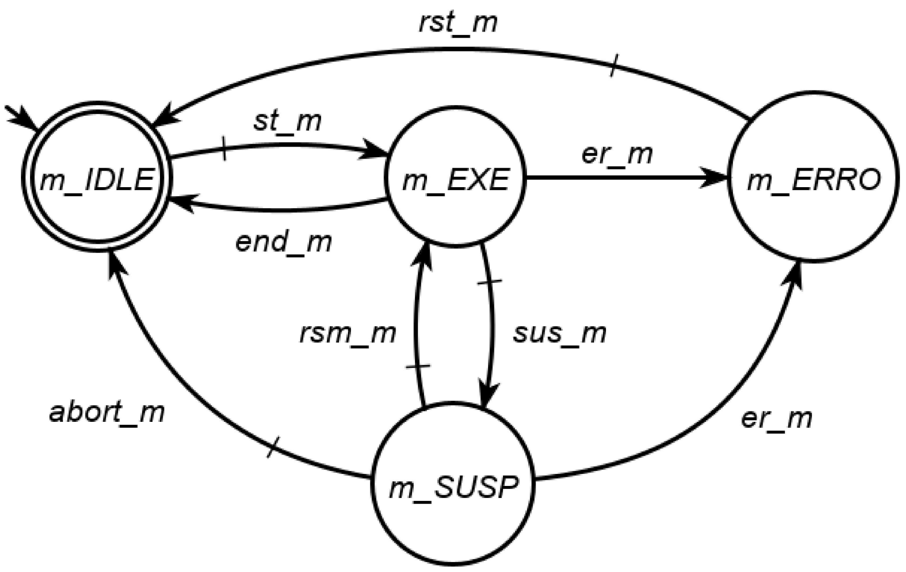
As we deal with the high-level control of maneuvers, the models proposed for these do not cover all the steps necessary for their execution, but only the transitions that directly affect their execution status. We propose the model of Figure 9 for maneuvers $m \in M$, being $M = \{app, assess, exp, rb, vsv, v\_search\}$, based on the approach proposed by Battistella (2015). By this model the maneuver m can assume four different operating states $\boldsymbol{S = \{m\_IDLE, m\_EXE, m\_SUSP, m\_ERRO\}}$. These represent that the maneuver has not been activated, that it is being executed, momentarily suspended or facing a failure, respectively. Suspend mode differs from idle in that the system maintains information necessary for the further continuation of the mission.

Robots are capable of starting, suspending, and aborting maneuver execution through controllable events, and the accomplishment of the maneuver or eventual errors trigger the ***end_m*** or ***er_m*** non-controllable events. All maneuvers dynamics are defined according to the events described in Table 8.

Table 8 – Definition of the events that determine maneuver dynamic.

Event	Type	Description
st_m	controllable	Maneuver m execution start
sus_m	controllable	Maneuver m is suspended, being ready for fast resume
rsm_m	controllable	The maneuver is resumed from the moment it was interrupted
abort_m	controllable	Maneuver is aborted, erasing all last target information
rst_m	controllable	Maneuver system is restarted from failures
er_m	non-controllable	A failure occurred on the maneuver system
end_m	non-controllable	Maneuver has been correctly executed

Figure 9 – Automaton G_m, which models most of the maneuvers executed by the robots in a high-level approach. Instead of considering all the components required to execute the maneuver, only events that change its current status are considered.



5.3.2 Subsystem 2: Teleoperation and Safe Land maneuvers

The Teleoperation and Safe Land maneuvers have some peculiarities that impose the need for a special model. Since teleoperation represents the exchange of the robot control from an autonomous mode to a mode in which humans can directly control its locomotion, the model G_{tele} presented in Figure 10 does not have a suspended mode. It would leave the robot lazy while humans were not available to perform its control. The safe land maneuver G_{land}, depicted in Figure 11, is very similar to the teleoperation maneuver, though, as it is specially designed for critical conditions, there is no need for an *abort* event, since the robot has to finish the maneuver at any cost.

Figure 10 – G_{tele} is an automaton that simulates teleoperation maneuvers. Observe that it does not have a suspended state since humans can only execute it or not. The robot must attain to autonomous tasks while teleoperation is not required.

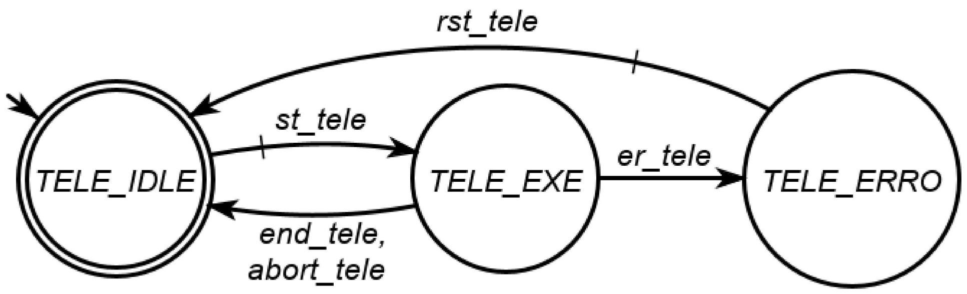
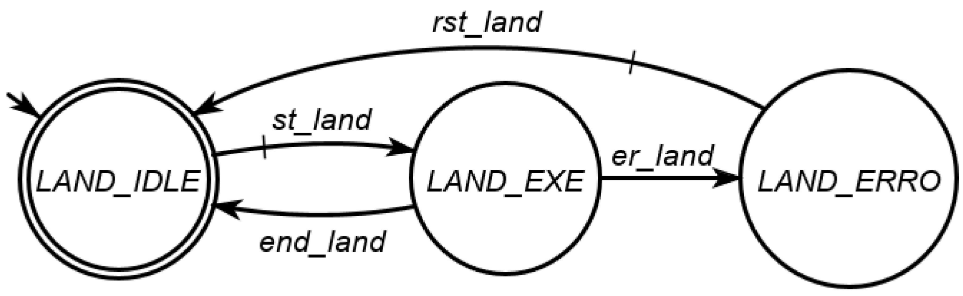


Figure 11 – G_{land} models the UAVs execution of landing following safe procedures, which are required when it is forced to land into the worksite.



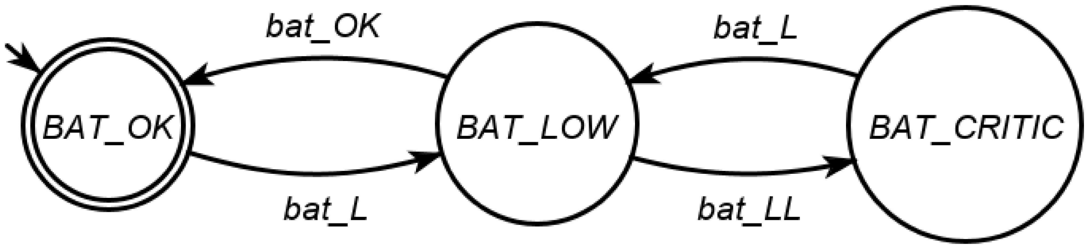
5.3.3 Subsystem 3: Battery consumption monitoring

The battery consumption monitor was modeled by the automaton in Figure 12, in which three operating states ***S* = {*BAT_OK*,*BAT_LOW*,*BAT_CRITIC*}** are considered, respectively representing: the battery under normal use conditions; the battery at a low level; and the battery at a critical level. To monitor the current state of this subsystem, three non-controllable events are used that represent changes in battery levels as described in Table 9.

Table 9 – Definition of the events that abstract the battery consumption level.

Event	Type	Description
bat_OK	non-controllable	Enough battery level for all functionalities execution (> 60%)
bat_L	non-controllable	Low battery level, indicating need of recharge (< 25%)
bat_LL	non-controllable	Insufficient battery for any task execution (< 10%)

Figure 12 – The system responsible for monitoring the battery level was modeled by the automaton G_{bm} composed of three states useful for delimiting operational modes for the robot.



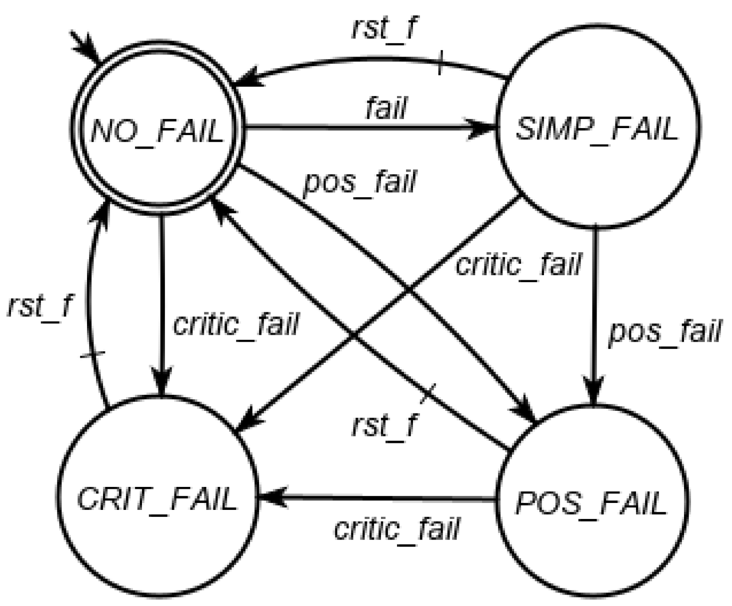
5.3.4 Subsystem 4: General failures

In addition to possible failures related to the incorrect execution of maneuvers, the system may have its operation affected by general failures of the location system and actuators responsible for the correct movement of the robot. To represent the system status related to failures on system components, we propose the model in Figure 13, which is made up of three types of possible failures, as described in Table 10. Here we defined a normal mode of operation and three degraded modes by the set of states $\boldsymbol{S = \{NO\_FAIL, SIMPLE\_FAIL, POS\_FAIL, CRITIC\_FAIL\}}$.

Table 10 – Definition of the events that occur as a result of system failures.

Event	Type	Description
pos_fail	non-controllable	Localization system failure
fail	non-controllable	Simple failure where the robot can still execute movements by its own
critic_fail	non-controllable	Failures that make robots movements unsafe
rst_f	non-controllable	Human command to signal that the failure was corrected

Figure 13 – The system responsible for monitoring the occurrence of general failures was modeled by the automaton G_{fail}, which changes the robot status according to the failure severity.



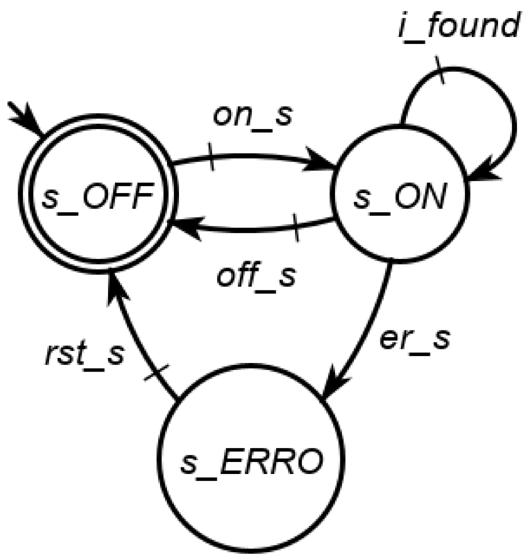
5.3.5 Subsystem 5: Sensors

In this work, two types of sensors were considered: the victim recognition sensor (VS) and the gas sensor (GS), and their operating conditions directly affect the execution of maneuvers. These were modeled according to the model shown in Figure 14, represented by three states in which each sensor is off (s_OFF), sensor s is on (s_ON) or sensor s is in a failure state (s_ERROR), for $s \in \{VS, GS\}$. As described in Table 11, it is possible to control the triggering of the sensors, but not their internal behavior.

Table 11 – Definition of the events generated by the sensors.

Event	Type	Description
on_s	controllable	Sensor ‘s’ is turned on
off_s	controllable	Sensor ‘s’ is turned off
er_s	non-controllable	A failure occurred on sensor ‘s’
rst_s	non-controllable	The sensor ‘s’ have been fixed

Figure 14 – With the automaton G_S we modeled the s sensor status. Knowing the sensor status is important for determining if dependencies on sensors are fulfilled.



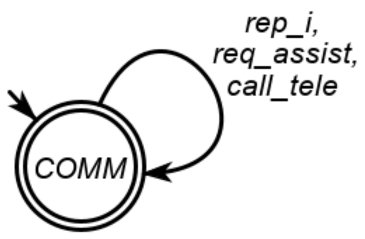
5.3.6 Subsystem 6: Communication

The interactions between humans and robots were represented by the model from Figure 15 composed of a single state and an auto-loop. This model was required to include the events of Table 12 to the alphabet of the plant, allowing the implementation of constraints interelating the robots behavior and their info exchange with humans. In this model we observe the events described in Table 12, which are related to requests and information exchange between robotic and human agents. Since the UGV is embedded with both types of sensors, consider that $i \in \{victim, gas\}$. But for the UAV, $i = victim$, as it can only recognize victims.

Table 12 – Definition of human-robot communication events.

Event	Type	Description
rep_gas	controllable	Report recognised gas leakage source position
rep_victim	controllable	Report found victim position
req_assist	controllable	The robot require a human to assist on the task execution
call_tele	non-controllable	A team member requires control of the robot movements

Figure 15 – With the automaton G_{com} we modeled information and command exchange between humans and robots.



5.4 SYSTEM SPECIFICATIONS

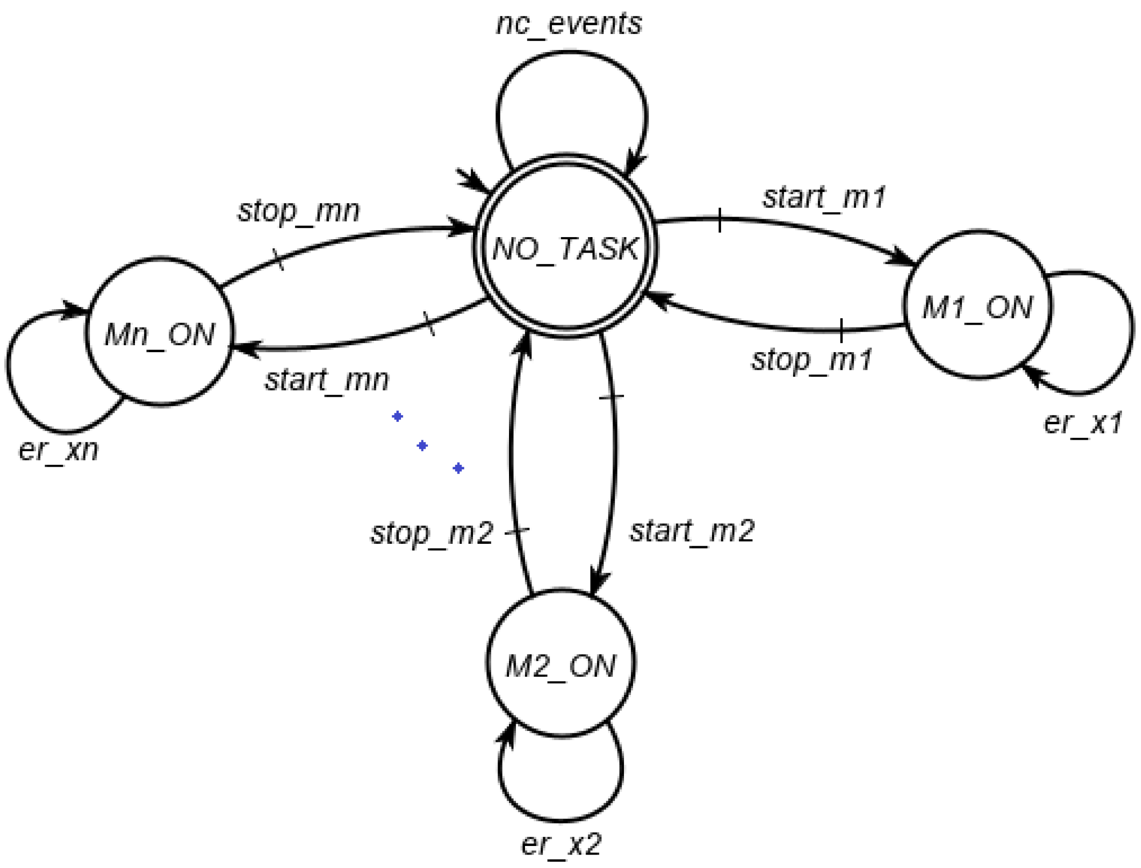
When robots are inserted into unknown and dynamic post-disaster environments, some relevant problems may arise, such as ensuring endurance (e.g., robots adapt to failures and other unexpected situations), safety (e.g., the actions of robots do not make the situation worse) and reliability due to the increase of systems based on artificial intelligence introduced to USAR. To ensure these characteristics and those highlighted in Section 5.2, a set of formal specifications were proposed and classified according to four main classes.

The proposed control system was developed to ensure the following projected specifications: 1. there are no conflicts in the use of shared payloads; 2. the robots maintain humans constantly updated about the new findings (Situational Awareness); 3. robots can deal with failures and act even with weakened systems; and 4. all dependencies of actions are met before their execution. In the sequence, we introduce an example of each of these classes, and the 15 specifications proposed for the UGV and 11 for the UAV are fully explained on Appendix A.

5.4.1 Subsystems conflicts prevention

All maneuvers of robots make use of the locomotion system, making their concurrent execution impossible. However, only the plant models do not prevent the mutual activation of maneuvers, so it is necessary to guarantee mutual exclusion through a specification. This is accomplished by the model in Figure 16, in which the beginning of a maneuver is linked to the end of another one that is in execution.

Figure 16 – An example of a specification for mutual exclusion between maneuvers When a maneuver changes to execution mode, the automaton moves from the internal node to an external one where starting another maneuver is not allowed.



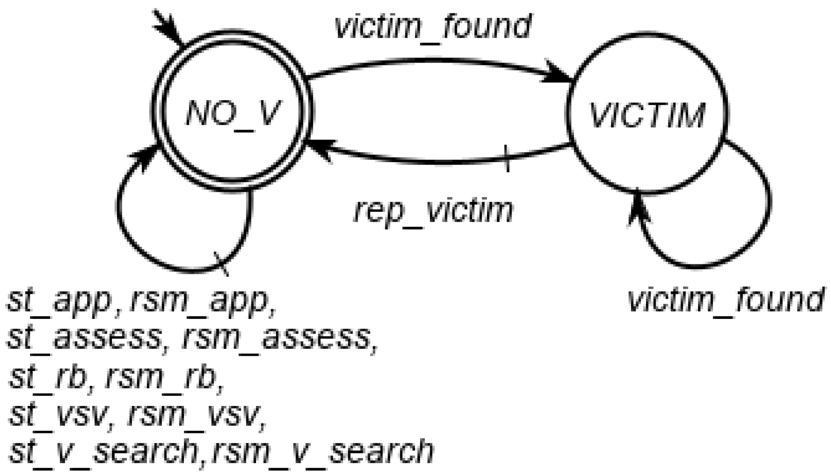
For better understanding, we have abstracted the real events by the ones described below. Observe that since the start events are not present on the peripheral states, they are not enabled, thus avoiding the mutual execution between maneuvers.

- *nc_events* = all non-controllable events contained in the automaton alphabet;
- *start_mi* = all events that make maneuver *i* change to execution mode;
- *stop_mi* = all events that make maneuver *i* change to idle or suspended modes;
- *er_xi* = error events related to all except *i* maneuver.

5.4.2 SA Specifications

Keeping the situational awareness of the Task Force Leader is extremely important as it allows him to quickly redesign the search strategy to be carried out by the teams. So, the automaton from Figure 17 is responsible for avoiding the UAV permanently executing maneuvers without updating the humans about a new found victim. A similar specification was designed to ensure the commitment of robots in transmitting the finding of a gas leakage, which is described in Appendix A.

Figure 17 – Specification E_{vr} was implemented to ensure that UAVs prioritize the situational awareness of the Task Force Leader in the event of finding a victim.



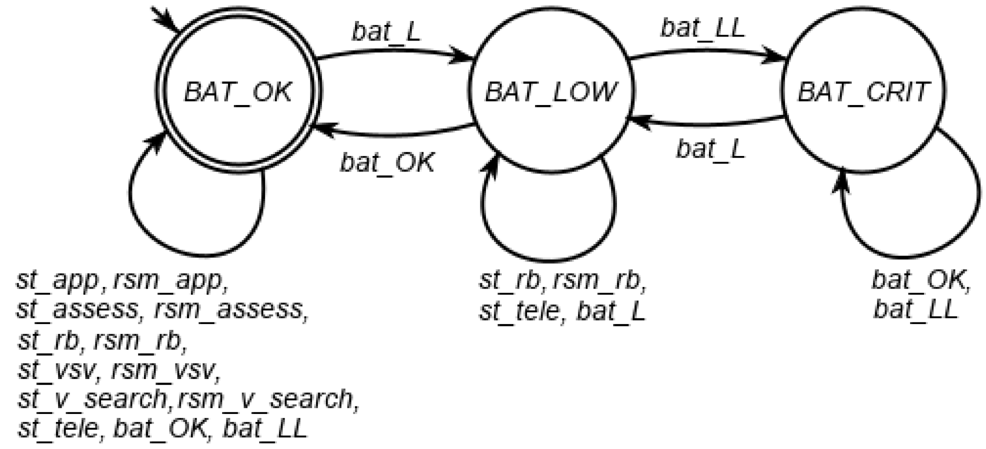
5.4.3 Robot applicability

To make the robots more robust to unexpected changes, we propose a set of degraded modes of operation according to the battery level and recognized failures. As an example, in Figure 18 we present the specification that defines the UAV allowed maneuvers in the face of different levels of battery. For each level, different sets of maneuvers are allowed, as follows:

1. ***BAT_OK*** **:** all maneuvers allowed since there is enough battery.
2. ***BAT_LOW*** **:** the *return to base* is allowed for recharging and *teleoperation* because a humans must be allowed to control the robot at any time.
3. ***BAT_CRITIC*** **:** all maneuvers enabled on BAT_OK are blocked to preserve the remaining battery for transmitting the robot position for further recovery.

Since events related to Safe Land are not present in the automaton, and they are not even present in the language alphabet, this specification does not disable their occurrence. So, in all the states of this model, including BAT_CRITIC, the event *st_safe_land* would be accomplished.

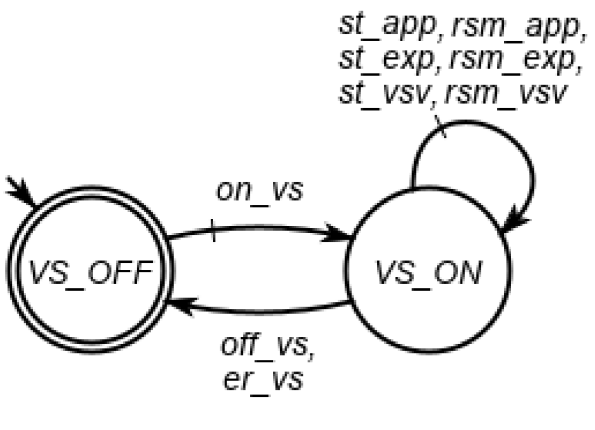
Figure 18 – Specification E_{bat} is responsible for delimiting the behavior of UAVs according to the current battery level.



5.4.4 Dependencies fulfilment

By this class of specifications, we avoid the start of maneuvers while dependent payloads are not in execution yet. As an example, the specification in Figure 19 ensures that the UGV only performs the approach, exploration, or surroundings verification if the victim recognition system is turned ON. With this, we ensure that the robot is alert to the presence of possible victims, and knowing their position it does not move too close, risking injuring them.

Figure 19 – Specification E_{v_dep} was created with the goal of disabling the start of maneuvers that require the victim recognition sensor to be turned ON in order to function properly.



5.5 SUPERVISORY CONTROL THEORY SYNTHESIS

The Supervisory Control Theory enables the synthesis of robust and minimally restrictive controllers based on formal languages. Though it presents a major limitation, the explosion of states. That is, overly complex models may result in large sets of states, rendering their real-world implementation impractical due to memory and processing requirements.

In the case of multi-robots systems, the state explosion problem might limit the scalability of robots used by the system. The Local Modular Supervisory Control Theory avoids such a situation by the synthesis of multiple non-conflicting supervisors and was therefore applied in the synthesis of supervisors in this work.

The supervisors were synthesised with the software Supremica (ROBI MALIK, 2021) and were proved to be non-confliting by symbolic methods, making possible the implementation of the LMSCT. Besides, the set of models related to the sub-systems and specifications have been proven to be non-blocking and controllable, so the supervisors can be represented by their respective specifications. Table 13 summarizes each supervisor synthesis procedure by means of states on the system. $E_{gen,x}$ and $G_{loc,x}$ represent the number of states in the specification and on the local plant (synchronization of all models with events present in $E_{gen,x}$). $K_{mod,x}$ is the states size of the modular specifications and $S_{mod,x}$ its respective modular supervisors. After reducing the supervisors, their final size is the one presented in column $S_{red,x}$.

Observe that it is possible to control the robots in a non-blocking and minimally restrictive manner without violating the controllability of events, with minimal supervisors feasible to be implemented in real systems. Instead of requiring the implementation of models composed of thousands of states, we have synthesised 15 supervisors with less than 7 states each for the UGV, and 11 supervisors for the UAV with up to 8 states each. As a result, in Chapter 7, we show how insignificant the computational cost required by the implemented formal models is.

Table 13 – The synthesis of local modular supervisors follows a procedure that generates a set of automata to convert a specification and the plants to supervisors. Here are all the automata generated in this process.

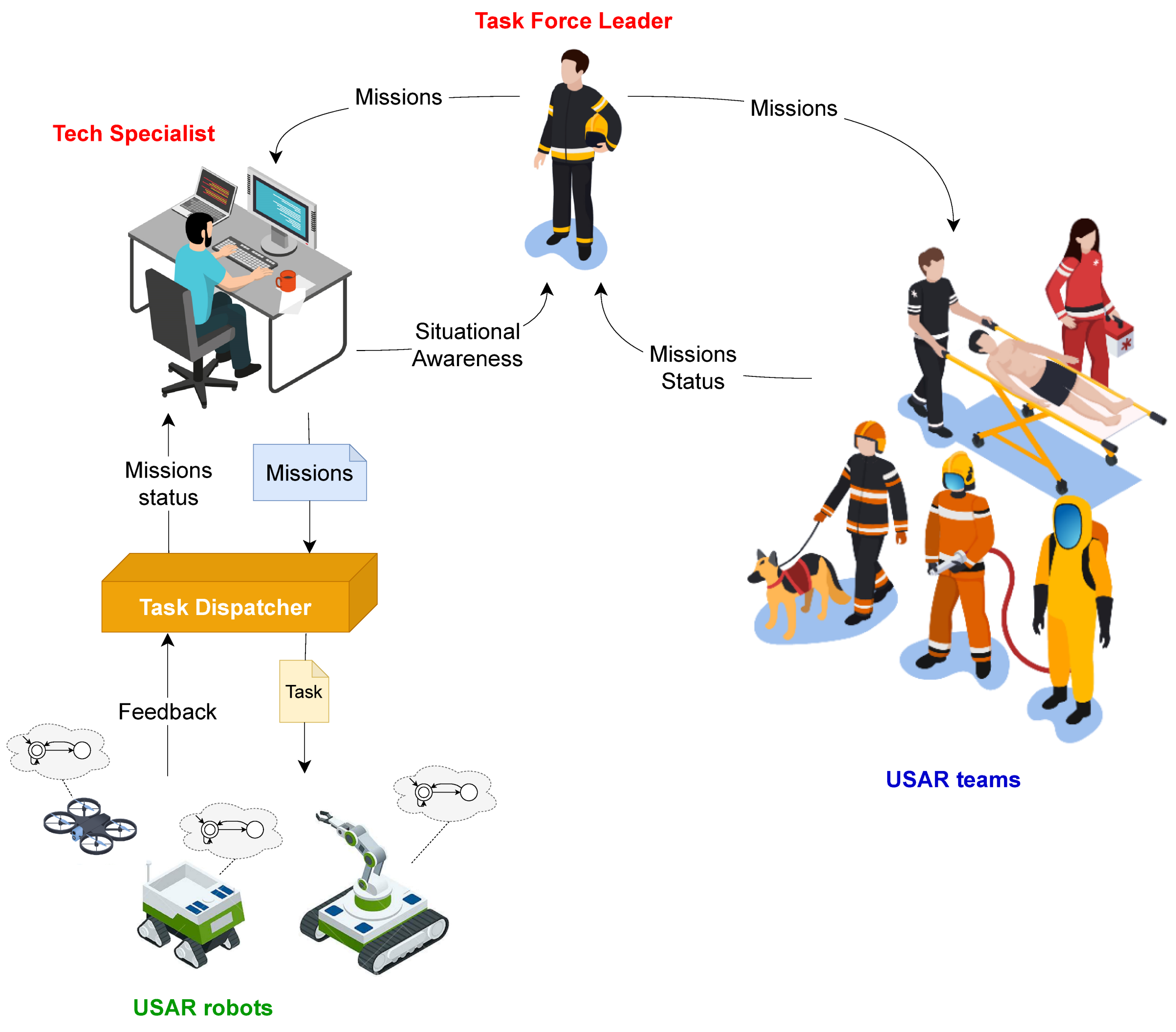
Robot	**Modular Supervisor** x	$E_{gen,x}$	$G_{loc,x}$	$K_{mod,x}$	$S_{mod,x}$	$S_{red,x}$
UGV	Mutual exclusion (S_{me})	6	768	459	459	**6**
	Victim sensor dependencies (S_{v_dep})	2	192	192	192	**2**
	Gas sensor dependencies (S_{g_dep})	2	48	48	48	**2**
	Victim sensors turn OFF condition (S_{v_off})	2	192	300	300	**2**
	Gas sensors turn OFF condition (S_{g_off})	2	48	66	66	**2**
	Assistance request condition (S_{assist})	2	1024	2048	2048	**2**
	Teleoperation prioritization (S_{tele_p})	2	768	1536	1536	**2**
	VSV start condition (S_{vsv_st})	2	12	15	15	**2**
	Victims report condition (S_{vr})	2	768	1536	1536	**2**
	Gas report condition (S_{gr})	2	768	1536	1536	**2**
	Simple failure operation mode (S_{s_fail})	2	256	384	384	**2**
	Position failure operation mode (S_{p_fail})	2	1024	1280	1280	**2**
	Critic failure operation mode (S_{c_fail})	2	3072	3072	3072	**2**
	Battery levels operation modes (S_{bat})	3	2304	2304	2304	**3**
	Sensors at critical battery ($S_{c\_sensors}$)	2	27	27	27	**2**
UAV	Mutual exclusion (S_{me})	8	9216	3564	3564	**8**
	Victim sensor dependencies (S_{v_dep})	2	144	144	144	**2**
	Victim sensors turn OFF condition (S_{v_off})	2	144	231	231	**2**
	Assistance request condition (S_{assist})	2	1024	2048	2048	**2**
	Teleoperation prioritization (S_{tele_p})	2	3072	6144	6144	**2**
	VSV start condition (S_{vsv_st})	2	12	15	15	**2**
	Victims report condition (S_{vr})	2	3072	6144	6144	**2**
	Simple failure operation mode (S_{s_fail})	2	1024	1536	1536	**2**
	Position failure operation mode (S_{p_fail})	2	4096	5120	5120	**2**
	Critic failure operation mode (S_{c_fail})	2	12288	12288	12288	**2**
	Battery levels operation modes (S_{bat})	3	9216	9216	9216	**3**

6 MULTI-ROBOTS DELIBERATIVE-REACTIVE COORDINATION SYSTEM

This chapter introduces the proposed deliberative-reactive Multi-robots Coordination System (MCS). First, we present the concept and explain each of the required layers to ensure the correct system behavior. Then, we summarize how the proposed architecture was implemented in a simulated environment developed on the Robotic Operating System (ROS).

The command hierarchy highlighted in Figure 20 represents how the proposed system fits with the human USAR teams, based on the procedures and chain of command proposed by the INSARAG. Here, a Task Force Leader defines missions (sets of sequential tasks) to be accomplished by human and robotic teams. A Tech Specialist

Figure 20 – Command hierarchy implemented in the human-robot USAR team. The MCS is composed by the robots and a centralized system that accomplishes the interface between them and the Tech Specialist.



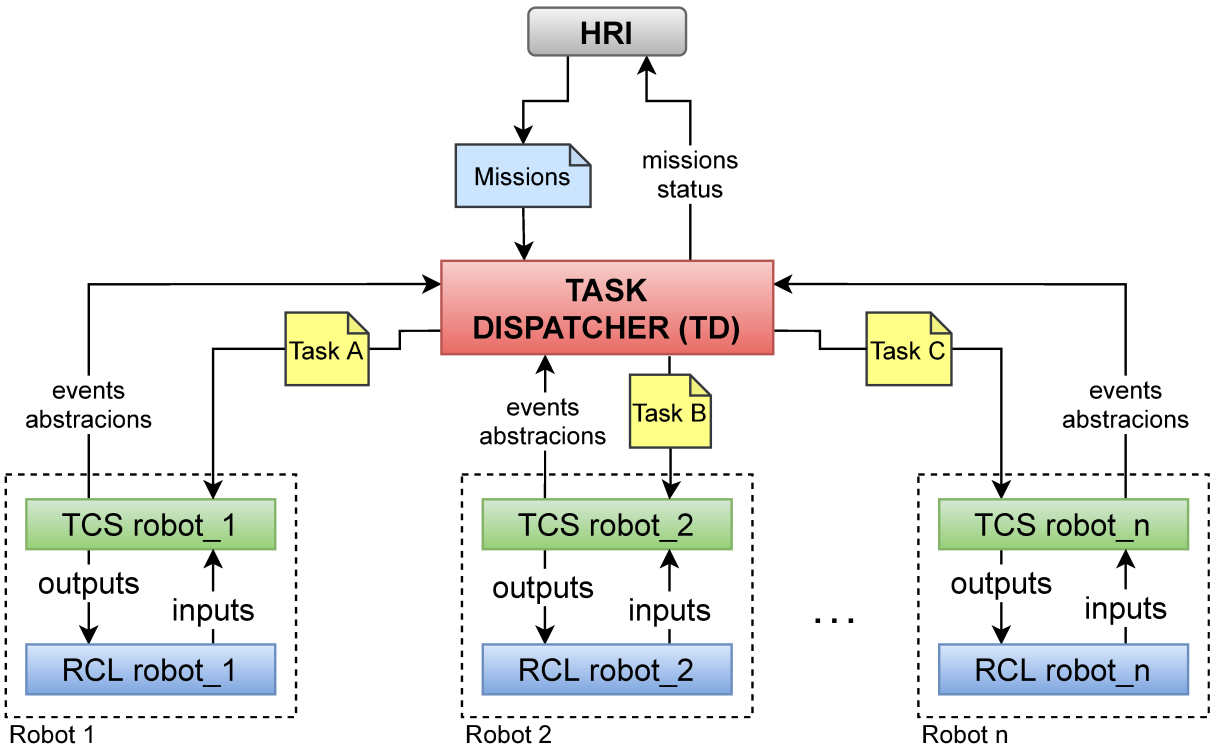
Source: Vectors available at https://br.freepik.com/vetores is then responsible for monitoring and interacting with the team of robots, and acting as an interface between the Task Force Leader and the MCS.

A centralized layer called Task Dispatcher (TD) is implemented in the MCS to reduce the human efforts needed to control a group of robots executing USAR tasks. As a result, the scalability of the system is not dependent on the monitoring capability of the leader. To ensure the reliability and safety of the actions executed, each robot has its own internal control system based on the supervisors synthesised according to the SCT.

6.1 MCS ARCHITECTURE

The proposed Multi-robots Coordination System follows the architecture shown in Figure 21. Missions are assigned by the Tech Specialist to robots through the Human Robot Interface (HRI), which interacts directly with the Task Dispatcher. The TD extracts the next task of each mission, monitors the current state of each robot, and assigns the tasks to the robots according to a search algorithm further explained in Section 6.1.1.

Figure 21 – Global view of the proposed architecture. The reactive system responsible for the safety and reliability of the system is embedded in each robot, and their coordination is accomplished by the TD, which assigns the tasks according to an A* search algorithm.



Each robot is composed of a Task Control System (TCS) which implements the Supervisory Control System (SCS) structure suggested by Queiroz and Cury (2002b).

The TCS ensures the reactivity of the robot to modeled events and carries out the safest action according to the embedded supervisors. At the same time, it maintains the TD updated about the status of the robots to help it determine the best tasks assignments.

The real implementation of the actions is only accomplished by the Robot Control Level (RCL). It represents the low-level where the commands executed by the TCS are converted into motion and sensing. This is where all the plants modeled in Section 5.1 are executed and the interface with payloads is done.

Therefore, the following two main layers can be highlighted in the MCS. A **centralized allocation layer** composed by the Task Dispatcher with the goal of coordinating the tasks being executed by the team of robots. A **Embedded Control Layer** represented by the Task Control System embedded on each robot, which must ensure the reliable and safe behavior of robots.

6.1.1 Centralized Allocation Layer

A **mission** represents a restricted sequence of tasks that must be executed. To make the TD capable of prioritizing the most relevant missions, a priority value from 0 to 10 is assigned to each mission, meaning 0 the most urgent. All the tasks belonging to the same mission receive equal priority, and the TD will always prioritize the allocation of tasks that belong to the most urgent missions, i.e., those that received priority close to 0. Each **task** represents the execution of a maneuver according to some parameters, defined by the tuple[1] below.

$$\boldsymbol{\sigma_{ij} =< a, p, r, s_{vic}, s_{gas}, t >}$$

- $i \in \{1,2,...,m\}$: mission index;
- $j \in \{1,2,...,n\}$: task index;
- σ_{ij}: jth task from mission i;
- a: specific agent or any;
- p: task goal position and orientation, represented by $< x, y, z, \theta >$.
- r: region of a task, delimited by an origin $< x_0, y_0 >$ and area size $< \Delta x, \Delta y >$;
- s_{vic}: status of the victim recognition system (ON or OFF);
- s_{gas}: status of the gas sensor (ON or OFF);
- t: one of the following tasks = [‘approach’, ‘assessment’, ‘search’, ‘return_to_base’].

[1] The tuple is based in the task representation proposed by Battistella (2015), but considering the payloads and maneuvers proposed in this work.

Instead of directly defining the maneuver to be executed in the task, we abstracted the maneuvers from Table 7 into only 4 possible tasks. *Approach*, *return_to_base* and *assessment* trigger the maneuvers with respective names, being the assessment only executable by UAVs. In terms of the *search* task, if the selected robot is an UAV it executes the 'victim_search' maneuver and if it is an UGV the triggered maneuver is the 'exploration'. The remaining maneuvers are not considered TD inputs. Victim_surroundings_verification and safe_land are considered only in backup behaviors, and teleoperation can be required directly through the HRI, not requiring a new mission. For example, a mission requiring the verification of two points followed by a full exploration of an area can be specified by the following:

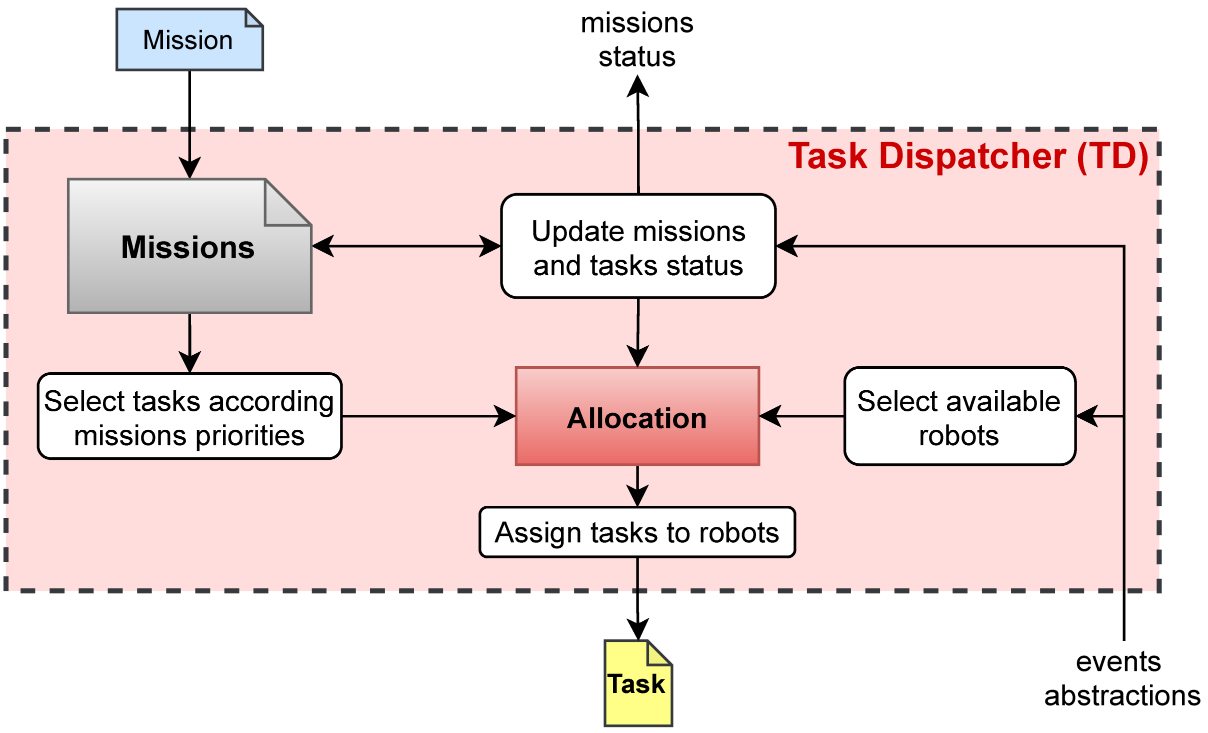
1. $< empty$,[16,16,3,0.7],$empty$,ON,OFF,$approach >$;
2. $< UAV2$,[32,34,5,0.7],$empty$,ON,OFF,$approach >$;
3. $< empty$,$empty$,[16,16,16,018],ON,OFF,$search >$.

Observe that the two first tasks have the fields 'position' filled and 'region' empty, which is the representation for objectives that have a point as a goal. On the other hand, in the third task we do the opposite since the required task might be executed in a region. Since tasks executed by the UGV do not require a Z position, the respective coordinate from 'position' is not considered if the ground robot is selected. If the intention is to force the robot to move to a higher position, away from the ground, it is possible by selecting which robot should accomplish the task, in this case an UAV like in task 2. If no robot is required, the TD selects as input of the allocation algorithm all available robots capable of executing the selected task.

The Task Dispatcher works online, so missions can be assigned or aborted at any time. For the correct allocation of tasks, the TD interfaces with all robots, from which it monitors: current status of tasks, possible failures, current position, and battery level. Then, as represented in Figure 22, it accomplishes the tasks reallocation considering the status of the current assigned tasks, available robots, and most urgent tasks required.

As the objective of the TD is to minimize the cost required to find the best robot to be allocated to each task, we have implemented a search tree algorithm. Each node is a set of tasks allocated to robots, and the branches represent a new robot-task pair allocated. For the tree expansion, we proposed as heuristic the cost function f, presented on Eq. (4). By f we intend to 1) prioritize higher-priority tasks; 2) reduce the reallocation of robots committed to higher priority tasks, 3) reduce the travelling distance; and 4) reduce the battery consumption of each robot. The higher the cost, the lower the chances of the robot-task pair being selected.

Figure 22 – Task Dispatcher process of task assignment to robots. The next required task of each mission and the available robots are selected, and the best $< task, robot >$ combination is obtained by a search algorithm.



$$f(r,t) = tp(r,t) + (10 - rctp(r)) + dist(r,t) + (100 - bat(r))/10 \qquad (4)$$

- $tp(r,t)$: priority of the task t being assigned to the robot r;
- $rctp(r,t)$: robot r current task t priority;
- $dist(r,t)$: euclidean distance between the robot r and the task t point;
- $bat(r)$: robot r battery level (%);

Algorithm 1 is basically an A* search algorithm where we verify, after each expansion, if all tasks have been allocated or if all robots are busy (line 6). If it is true, we return the current node n, that has the task allocated to each robot, and if not, we verify if the current branch is the longest one, meaning that it is the one with the most tasks assigned. To ensure that missions with higher priority are executed first, on line 11 we add as successors only tasks belonging to missions with higher priority, i.e., tasks with lower priorities are assigned as successors only if the tasks with higher priority have already been allocated or have no robot to execute them.

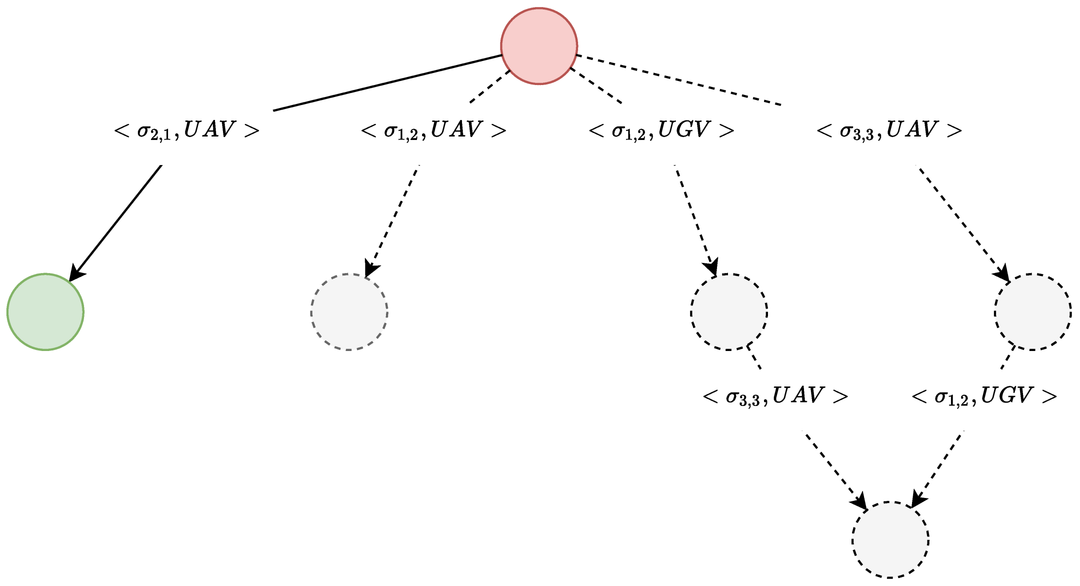
Suppose that at a certain moment the system has one UAV and one UGV available and three missions to be accomplished, m_1 (5) - m_2 (1) - m_3 (3), with the

Algorithm 1 Task allocation.

```
Input: robots(R) and tasks(Σ)
Output: task → r
 1: sort(Σ)
 2: PriorityQueue(f) open
 3: open.add(Node initial)
 4: while not open.isEmpty() do
 5:     Node n ← open.remove()
 6:     if all $\sigma$ or all r allocated then
 7:         return n
 8:     else if n.tasks > max_tasks_node.tasks then
 9:         max_tasks_node ← n
10:     end if
11:     open.add(n.successors())
12: end while
13: return max_tasks_node
```

number representing their respective priority. If the current tasks $\sigma_{1,2}$, $\sigma_{2,1}$ and $\sigma_{3,3}$ are respectively accomplishable by UAVs, UGVs and any robot, the expected tree of possibilities would be the one of Figure 23. The whole tree represents all possible compositions without considering only the highest priorities as successors. Observe that in this case, the system would allocate two of the three tasks by the paths to the right (dashed). But since the system is designed for assigning first the most immediate tasks, it prioritizes allocation of only $< \sigma_{2,1}, UAV >$ because it is the only possible way to assign the highest priority task.

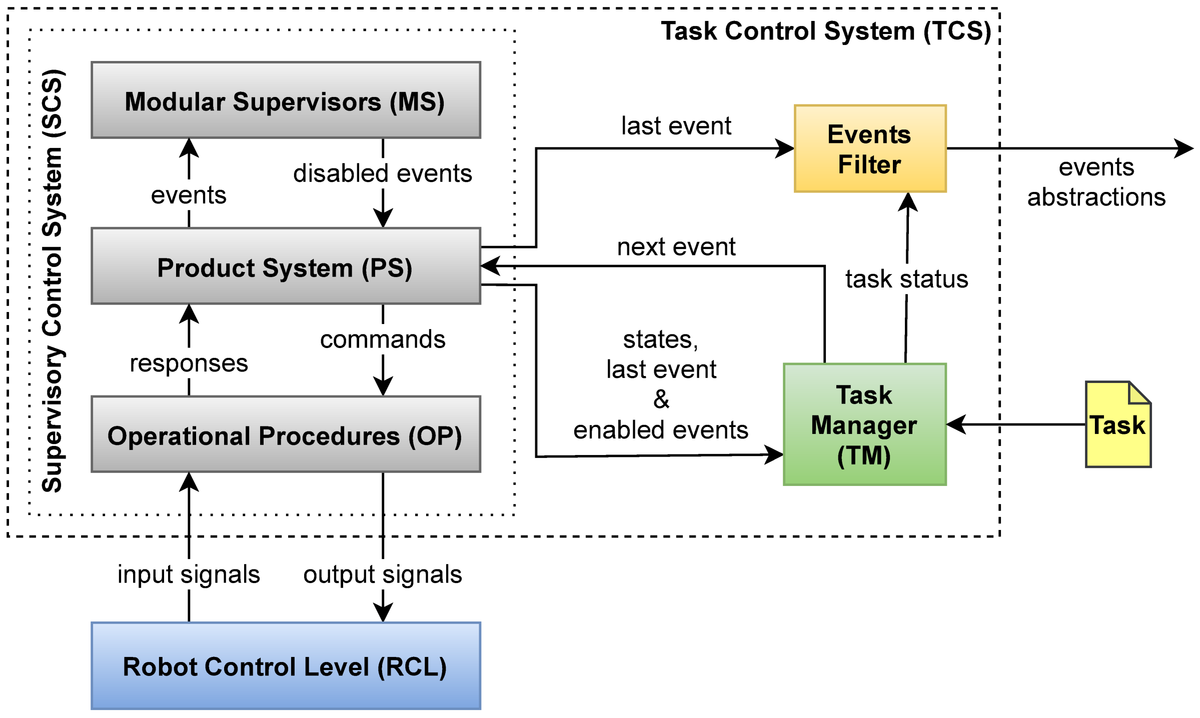
Figure 23 – Comparison of the possibilities tree with and without mission prioritization. If there was no mission prioritization, it would be possible to assign two tasks instead of only one.



6.1.2 Embedded Control Layer

The **Task Control System (TCS)** shown in Figure 24 is embedded on each robot with the purpose of controlling the next action according to the SCT. The TCS is based on the approach of Battistella (2015), which is composed of the Supervisory Control System (SCS) and a Mission Manager, which here we named Task Manager (TM) since the tasks assignment is the TD responsibility. To provide the interaction between TCS and the TD we added an Events Filter (EF) to it.

Figure 24 – The elements of the TCS interact with each other to ensure the behavior of the robot according to the modeled specifications and provide the needed interaction with the TD through abstracted events.



Source: Adapted from Battistella (2015).

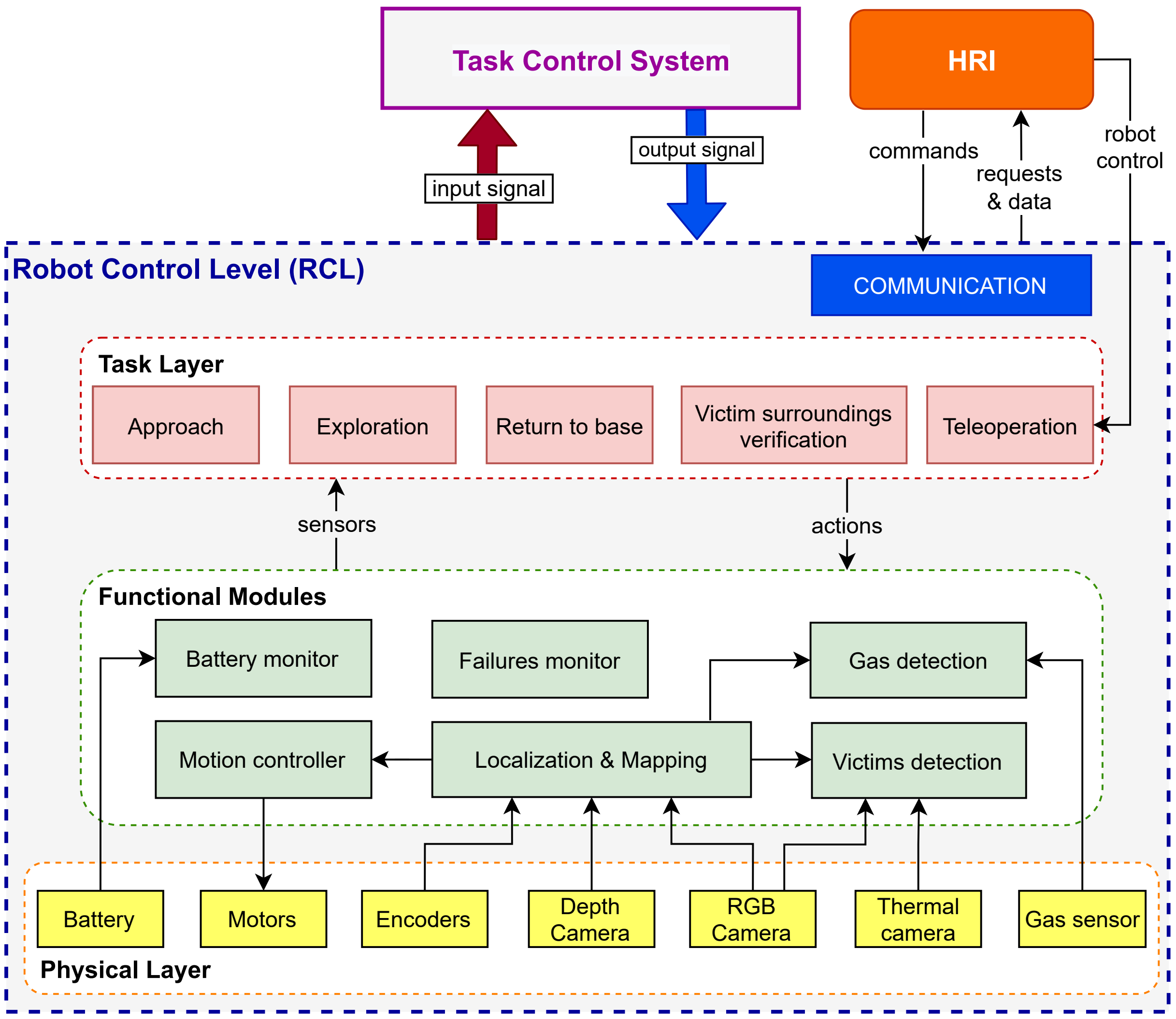
The **Supervisory Control System (SCS)**, first approached by Queiroz and Cury (2002b), is a three-level hierarchy that allows the implementation of the SCT on real systems. Here, models and supervisors are implemented through an interface between high-level events and low-level signals and commands recognized by the Robot Control Level (RCL). This translation between high-level events and low-level signals is accomplished by the **Operational Procedures (OP)** block. It can be a conversion of an analogical value to a discrete representation (e.g., battery supply percentage to operational levels), or a direct conversion from an event to a physical signal or sequence of signals.

To control the robots with the payloads and proposed behaviors described in

Section 5.1, the UGV ends up with 10 subplants (5 maneuvers + payloads), while the UAV is composed of 11 subsystems (7 maneuvers + payloads - gas sensor). These plants are implemented in the **Product System (PS)** and their dynamics are defined by the events forced by the Task Manager and by responses coming from the OP. By transmitting the occurred events and receiving the disabled events, the PS also interacts with the **Modular Supervisors (MS)**, which implement the supervisor synthesised in Section 5.5.

As previously mentioned, the proposed models only abstract the real system behavior and do not consider every step needed for the correct functioning of the subsystems. The RCL implements the real systems responsible for controlling each physical component of the robot. For example, in Figure 25 is shown the RCL of the implemented UGVs. Observe that it is composed of subsystems responsible for controlling the motion of the robot and its localization and mapping, though these are not

Figure 25 – The RCL of UGVs is composed by the modeled maneuvers and functional modules that interface with the physical components.

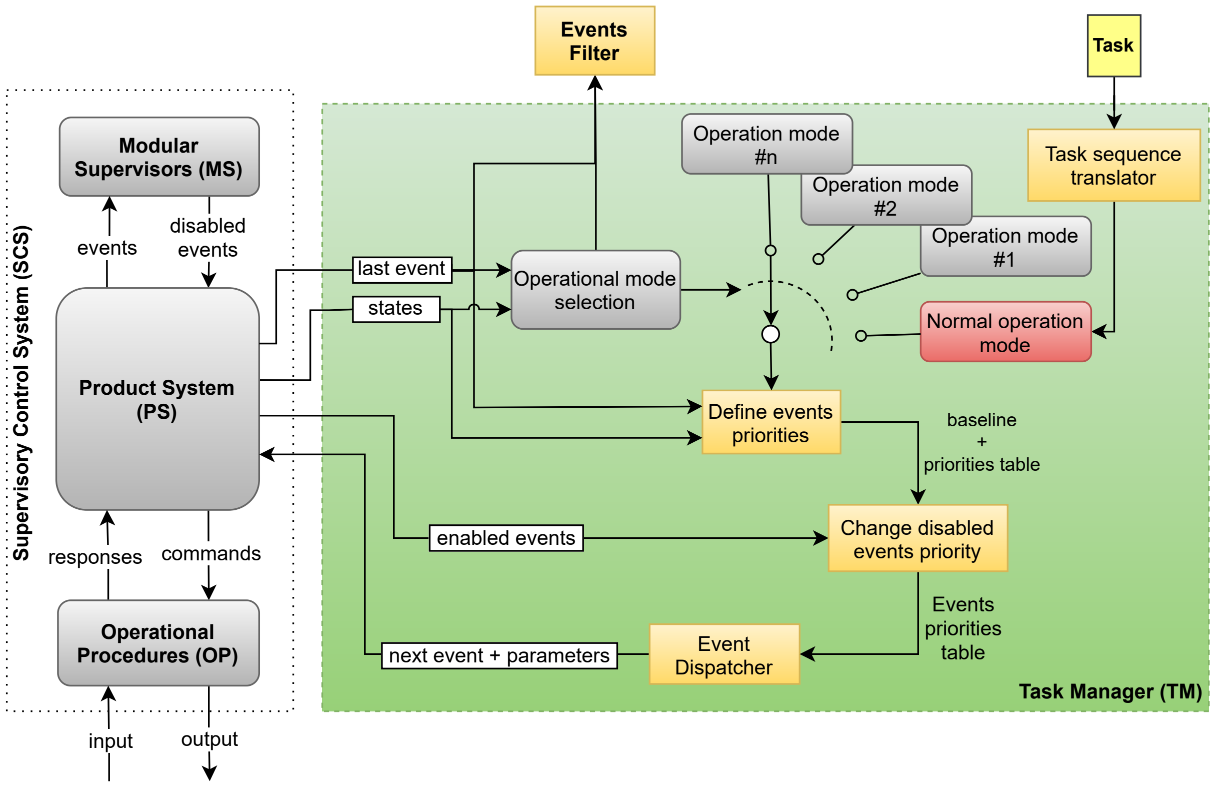


considered in the models. The implementation of high-level models simplifies the system and does not restrict the architecture to a specific control of the functional modules. As long as the functional module can be fully controlled by the proposed events, it becomes applicable to the proposed system.

The **Task Manager** is also very important to the implementation of the TCS. It is responsible for selecting the robot mode of operation and executing only those events enabled by the SCS. As can be seen in Figure 26, the TM selects if the task assigned by the TD must be executed (normal operation mode) or if the robot should accomplish a backup behavior, e.g., returning to base due to low battery levels.

When there is a change in the Product System or a new task arrives, the TM selects the most appropriate mode of operation and assigns priorities to all modeled events. Each mode of operation has an events priority table that assigns greater weight to the most desired events. It also defines baseline events based on the current state of the system. The events priority table of a mode of operation is static, while the baseline can change due to changes in the system state.

Figure 26 – The Task Manager controls the behavior of the robot based on its current state, events enabled by the supervisors, and tasks assigned by the TD.



Source: Adapted from Battistella (2015).

All baseline events are assigned with a priority value of 10. And the priority table is defined based on the desired behavior of the current mode of operation, being possible to define values bigger or smaller than the baseline. For example, since we would like to ensure the Task Force Leader situational awareness, events like **report_victim** and **report_gas**, receive a priority higher than 10. So the system would always prefer to execute the report event instead of a baseline event if it is enabled by the SCS.

The controllable events disabled by the SCS receive null priority, even if they are required by the current selected mode of operation. Then the TM selects from the priorities table the highest one. If it is non-controllable, the TM can only wait for new events, but if it is controllable, the TM can execute the event by sending it to the SCS, with the required parameters for its correct execution. To avoid system blocking and ensure that a disabled event is not executed, the minimum priority assigned to non-controllable events is 1. And since disabled events receive null priority, the system would prioritize the waiting for non-controllable events occurrence instead of executing a disabled controllable event.

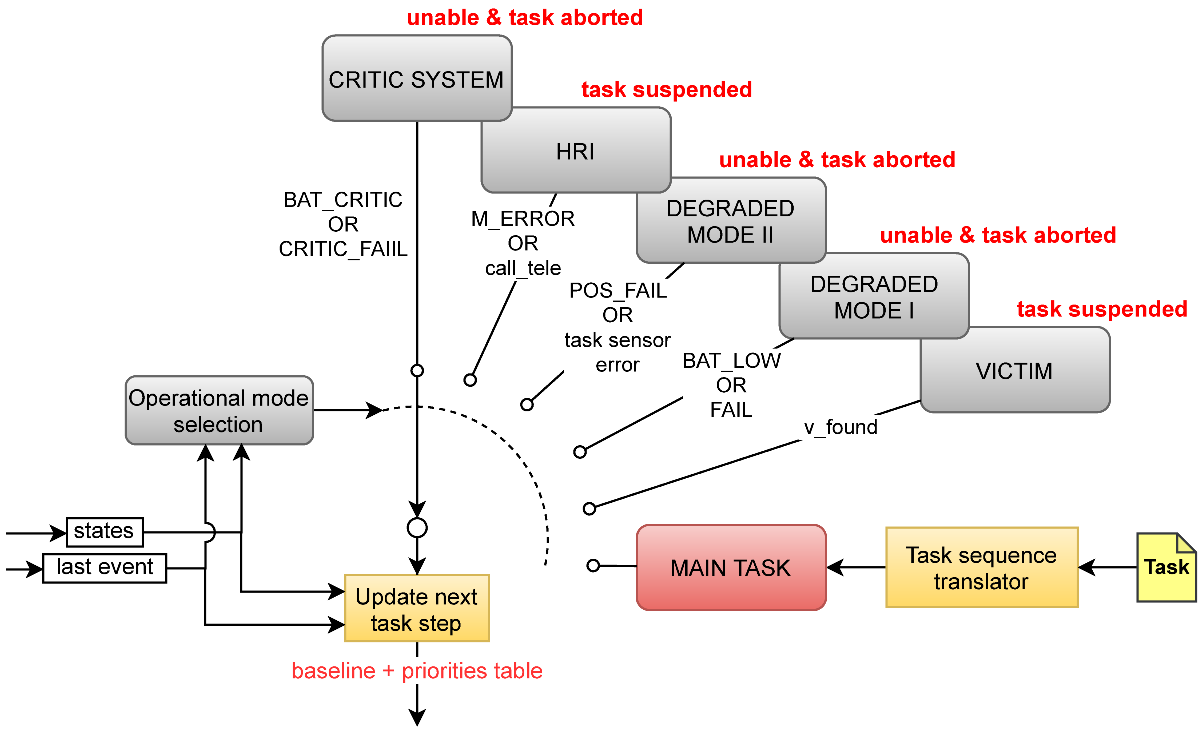
UAVs and UGVs have been proposed to have similar, but not identical, behaviors. Each robot can operate in the normal mode or in one of the five alternative modes. While the system is in normal operation conditions, the TM selects the task assigned by the TD and the **Task Sequence Translator** converts the tuple into an events baseline and a table of priorities, which are built based on the required sensors and selected task. The baseline ensures that sensors required by the task or by the specifications are turned on, followed by the maneuver execution and sensors deactivation, while the priority table prioritises the finalization of maneuvers and notification of findings to maintain SA. For example, if the robot receives a 'search' task and it requires no sensor to be turned on, the translator will still attempt to turn on the sensors before executing the maneuver, because it is required by the models as a precondition for the maneuver to start.

The modes of operation of both UAVs and UGVs are presented in Figure 27. The priority of the behaviors decreases from top to bottom. It means that the system will preempt the current behavior if a condition that would start a higher priority mode is satisfied, e.g., if the CRITIC SYSTEM mode is satisfied while HRI is currently selected. The conditions that trigger new modes of operation are shown in Table 14.

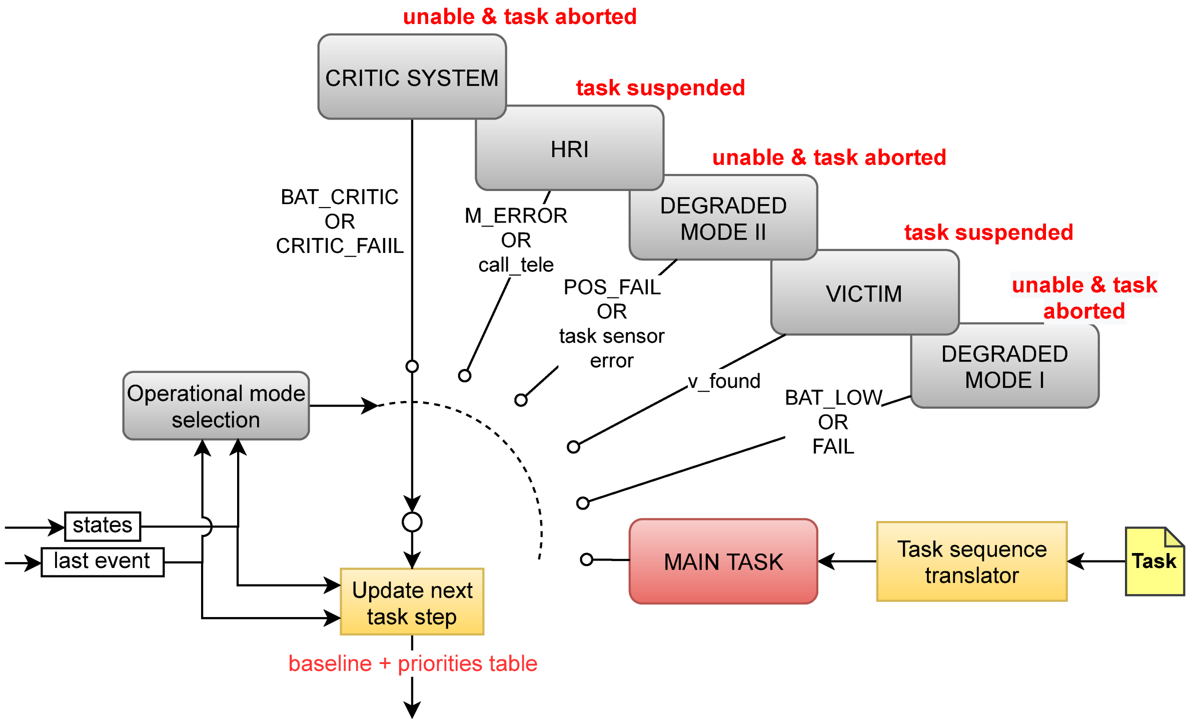
Table 14 summarizes the goal of the operation modes, and a more deep explanation is presented on Appendix B. Each operation mode also defines a status for the robot and for the main task, which is transmitted to the TD by the Events Filter. The robot can assume the status IDLE, BUSY or UNABLE. And tasks are classified as EXECUTING, SUSPENDED, FINISHED or ABORTED. When a change occurs to the robot or task status, it is sent to the TD. Table 14 also describe the changes in such status when backup behaviors are triggered.

As the last component of the TCS, we propose the **Events Filter** as the component responsible for updating the TD about the current status of the robots. Rather than sending all of the events that occur on each robot to the TD, the EF monitors the robot

Figure 27 – UAVs and UGVs behaviors and their triggering conditions. The main difference between UAVs and UGVs is the priority of the behavior triggered by finding a victim.



(a) UAV modes of operation.



(b) UGV modes of operation.

Table 14 – Proposed modes of operation. Each behavior was proposed with an objective and is triggered by specific conditions. When assuming a new operation mode, the TD is updated with the task and robot status.

Behavior	Objective	Trigger condition	Status send to TD
CRITIC SYSTEM	Abort all maneuvers and turn off all sensors. *The UAV first have to execute safe land avoiding to land to close to victims	Critic battery level or critic failure	Robot unable to execute task and main task aborted
HRI	Execute teleoperation of the robot by requisition of the human or by the robot	Maneuver error or human call teleoperation	Main task is suspended
DEGRADED MODE II	Non essential maneuvers are aborted since the robot can not locate itself	Position failure or failure of a sensor required by the main task	Robot unable to execute task and main task aborted
DEGRADED MODE I	Return the robot to base. UAV executes it immediately and UGV only after the current task is accomplished	Battery level is low or a simple failure occurs	Robot unable and main task aborted (only for UAV)
VICTIM	Execution of the victim surroundings verification	A victim is found	Main task is suspended

dynamics and abstracts the changes into a smaller set of events. Sending all events would require the TD to implement all the robots plants and would increase the amount messages exchanged.

Since the cost function implemented on TD depends on the status of tasks and the robots selection is based on the status of robots and sensors, the EF monitors errors in the sensors, robot status changes, and tasks changes due to a new mode of operation selected or task finalization. The battery level and robot position, also required by the TD, are transmitted directly from the RCL to TD, without passing on the TCS.

The Events Filter constantly monitors the current status of the robot and of the assigned task and the occurrence of failures in the sensors. If, for example, the robot is currently executing the main task with the status BUSY and EXECUTING and it finds a victim, a change to the VICTIM behavior is expected, which would cause the new task status to be SUSPENDED. When the EF detects this change, it immediately notifies the TD. Now suppose that in a second moment it faces a *critical_failure* leading the robot to be UNABLE and the task ABORTED. The abstractions sent to the TD would cause it it replan the allocation of the tasks, ignoring the unable robot.

Not all the abstractions lead to a replanning of the tasks allocation. The TD will replan in the following conditions: a robot finishes a task (FINISHED); a task is ABORTED by the robot; a robot gets IDLE; a sensor in failures is reestablished; and, of course, a new mission is inserted or an old one is removed by the HRI. The abstractions that do not trigger the TD help it knowing the current status of robots and tasks, saving all changes that have occurred in the robots. In this work, it was important to help us evaluate the behavior of the system, working as a datalog. In the implementation of robots in USAR field situations, it would be fundamental for the process of lessons-learned review, phase in which the USAR teams evaluate the executed procedures in order to improve the overall effectiveness and efficiency for future disasters response.

6.2 MCS IMPLEMENTATION

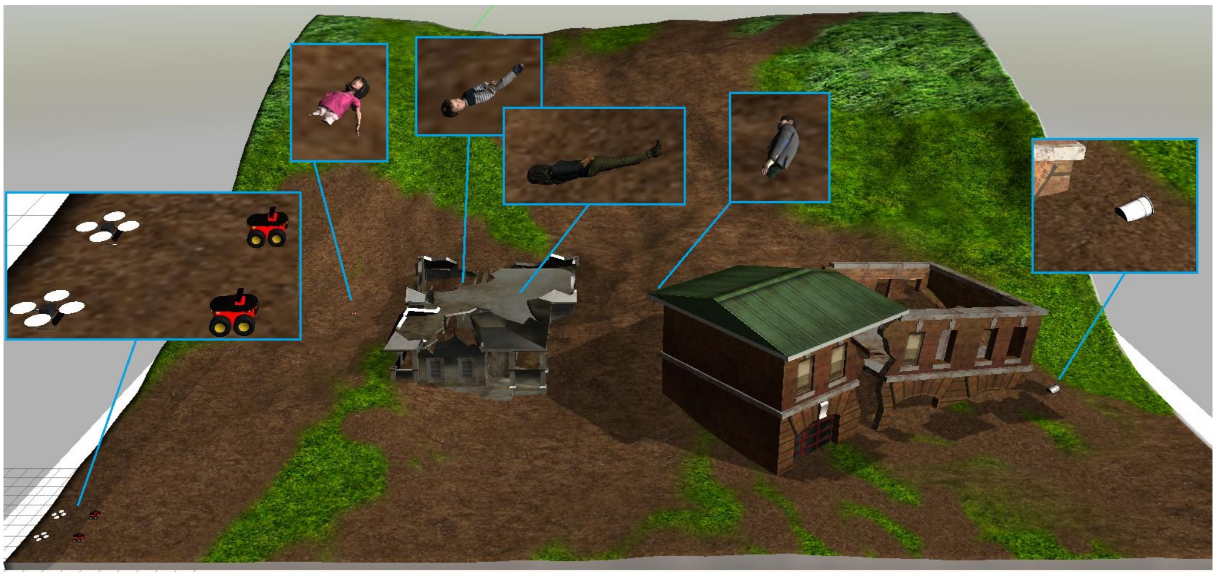
To validate the proposed architecture, we have implemented it in a simulated environment fully developed in the Robotic Operating System. It is an open source ecosystem composed of a set of software libraries and tools specially designed for robot applications. ROS basically works as a distributed environment where systems developed in different languages can interact in a publish-subscriber communication interface through elements called **topics**. When a **node** (ROS element) publishes a message to a topic, all nodes subscribed to this topic receive the message.

With the implementation of the proposed scenario, we expect to validate the robustness of the SCS embedded in each robot. It was designed in a way that some relevant concepts of USAR situations were present, e.g., the presence of victims, gas, and semi-collapsed structures. But the physical distribution was attenuated to make possible the use of robots with a broad set of packages available in ROS, e.g., the **pioneer3at** and **hector_quadrotor** robots were implemented due to the available packages to control the low-level systems. Although these types of robots are not suitable to be applied to real USAR situations due to their inefficient locomotion capabilities, their use in a simulated environment is enough when the objective is to verify the behavior of robots in an unknown environment and in the face of the occurrence of unexpected and non-controllable events.

The 3D simulated post-disaster environment shown in Figure 28 was developed using the Gazebo framework, designed for robotic simulations. It makes possible not just to implement a visual 3D environment, but to simulate physical behaviors and interactions between the robots applied to it. We proposed a landslide scene composed of two semi-collapsed structures that could be dangerous for humans, but still accessible to robots. Four partially buried victims (2 on the exterior and 2 on the interior of the house) and 2 gas sources (next to the bigger building) were placed along the scene. And as mentioned in Chapter 5, two Pioneer3at (UGV) and two drones (UAV) were inserted on the environment to execute the tasks assigned by humans.

All low-level operational systems of robots responsible for mapping, trajectory planning, collision avoidance, and motors control were implemented through well established ROS packages. The proposed maneuvers were implemented by making use

Figure 28 – Simulated landslide environment and respective distribution of buildings, victims and dangers.



of some existing features applied to packages designed by us. And to ensure that the maneuvers follow the principle of operation proposed for the models, each robot has a controller responsible for monitoring the output commands from the OP and converting them into action execution. The sensors and failures and battery monitors were specially implemented by us just to easily simulate the full system. So it might not be considered as an attempt to increase efficiency.

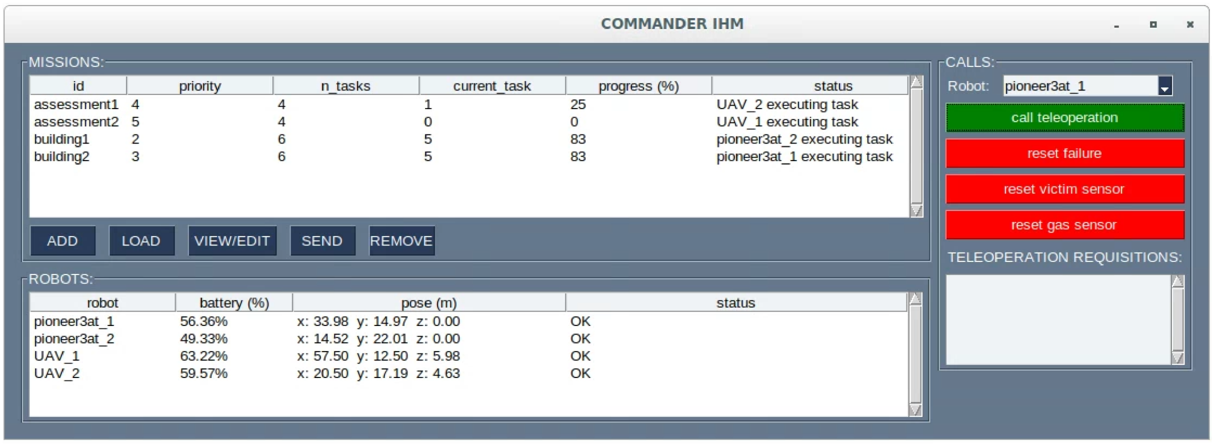
The proposed architecture was fully implemented in Python and a translation system from Automata to Python language was developed to make possible to easily update the Supervisory Control System embedded in the robots. For a better understanding about the translation system and how the SCS was implemented, refer to SIMON (2021a) or to the brief explanation present in Appendix C.

Two Graphical User Interfaces (GUI) were developed, the one showed in Figure 29 acts as the HRI highlighted in Figure 21. In which it is possible to create and delete missions, send them to the TD and monitor the position, battery level, and failure status of all robots. Through this interface the human is also capable of requiring teleoperation of robots and resetting failures. The visualization of maps generated by the robots and their camera images was not implemented directly on this interface, but can be accessed via Rviz, a 3D visualization tool for ROS.

The second interface, shown in Figure 30 was developed to help in monitoring the occurred events and the current state of robots. It is also capable of forcing the occurrence of non-controllable events to allow us to easily simulate and validate different behaviors of the robots in the face of unexpected situations.

All the features and implemented code needed to execute the MCS in this sim-

Figure 29 – Graphical Human Robot Interface designed for missions assignment and robots monitoring.

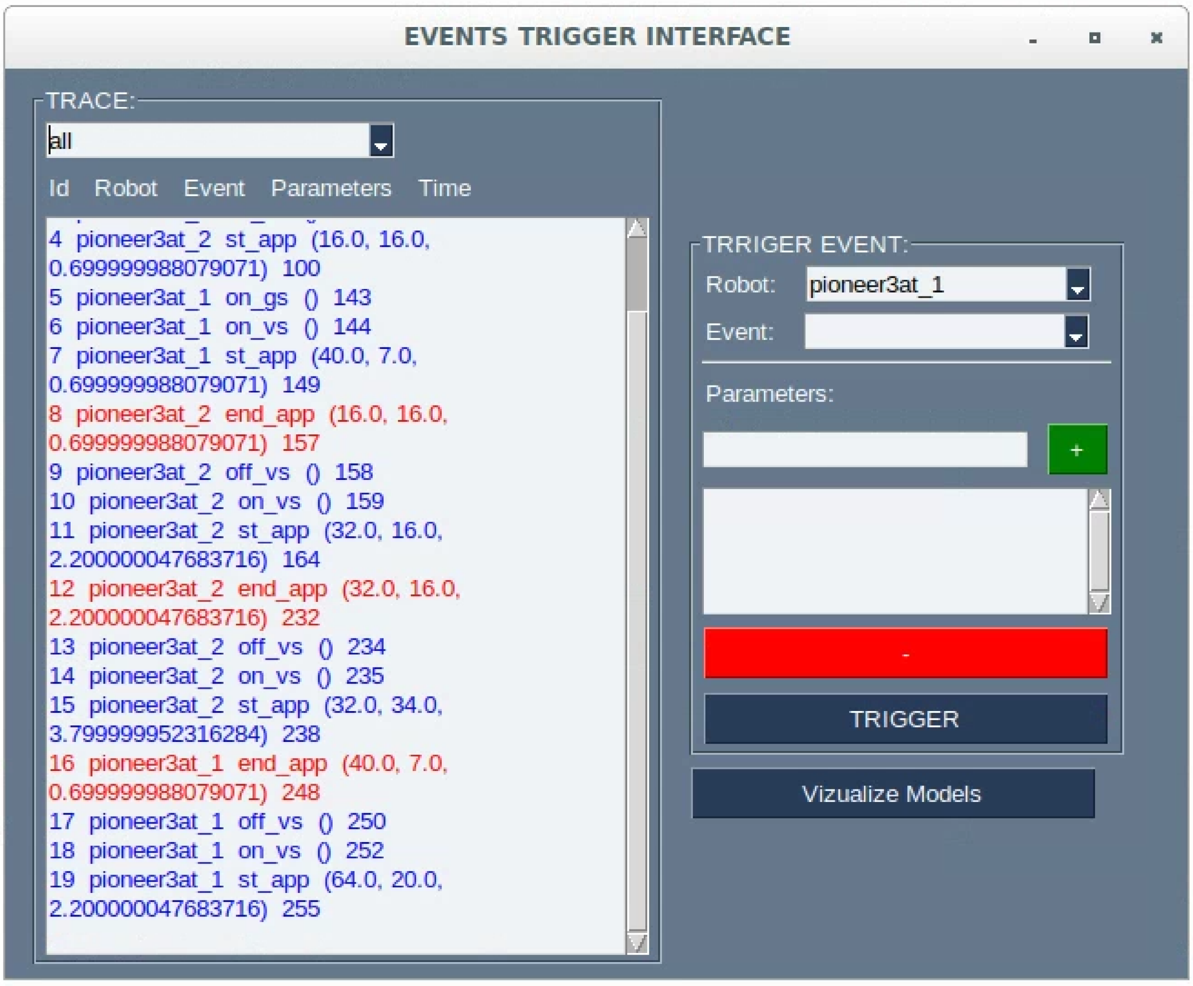


id	priority	n_tasks	current_task	progress (%)	status
assessment1	4	4	1	25	UAV_2 executing task
assessment2	5	4	0	0	UAV_1 executing task
building1	2	6	5	83	pioneer3at_2 executing task
building2	3	6	5	83	pioneer3at_1 executing task

robot	battery (%)	pose (m)	status
pioneer3at_1	56.36%	x: 33.98 y: 14.97 z: 0.00	OK
pioneer3at_2	49.33%	x: 14.52 y: 22.01 z: 0.00	OK
UAV_1	63.22%	x: 57.50 y: 12.50 z: 5.98	OK
UAV_2	59.57%	x: 20.50 y: 17.19 z: 4.63	OK

Robot: pioneer3at_1

Figure 30 – Interface designed to monitor events occurred in every robot and to force unexpected situations.



all

```
Id  Robot  Event  Parameters  Time
4  pioneer3at_2  st_app  (16.0, 16.0,
0.699999988079071)  100
5  pioneer3at_1  on_gs  ()  143
6  pioneer3at_1  on_vs  ()  144
7  pioneer3at_1  st_app  (40.0, 7.0,
0.699999988079071)  149
8  pioneer3at_2  end_app  (16.0, 16.0,
0.699999988079071)  157
9  pioneer3at_2  off_vs  ()  158
10  pioneer3at_2  on_vs  ()  159
11  pioneer3at_2  st_app  (32.0, 16.0,
2.200000047683716)  164
12  pioneer3at_2  end_app  (32.0, 16.0,
2.200000047683716)  232
13  pioneer3at_2  off_vs  ()  234
14  pioneer3at_2  on_vs  ()  235
15  pioneer3at_2  st_app  (32.0, 34.0,
3.799999952316284)  238
16  pioneer3at_1  end_app  (40.0, 7.0,
0.699999988079071)  248
17  pioneer3at_1  off_vs  ()  250
18  pioneer3at_1  on_vs  ()  252
19  pioneer3at_1  st_app  (64.0, 20.0,
2.200000047683716)  255
```

Robot: pioneer3at_1

ulated environment are available at SIMON (2021b). By correctly implementing it on ROS, it is possible to initially simulate the four proposed robots. But it would be easy to insert more robots as long as there is enough computational resources to do so. Although we have implemented some specific packages to accomplish the maneuvers execution, someone with considerable knowledge of ROS could implement other approaches without affecting the MCS architecture. Because we abstract the subsystems in a high-level representation, changes in how they work at the low-level cannot affect the reliability of supervisors.

7 SIMULATIONS AND RESULTS

The proposed architecture was evaluated in a simulated environment with the objective of validating that the SCT is feasible for USAR applications. We do not intend to prove that the formalism of the SCT is correct; it has already been proven by Ramadge and Wonham (1988) and Queiroz and Cury (2000). In this section, we focus on ensuring that the synthesised controllers were correctly implemented and that the robots reacted as expected in the face of uncontrollable events. Also, we verified the synergy between the centralized layer and the robots and the computational cost of implementing the architecture to see if it is feasible for embedded systems.

A set of missions were proposed to evaluate the system performance and the correct behavior of robots. These have been formulated as an attempt to represent procedures that follow the INSARAG guidelines. The missions intend to increase the knowledge about the environment, first in a superficial way (Levels 1 and 2 of Table 1), then in a more committed and focused approach (Levels 3 and 4 of Table 1).

The proposed missions A1 and A2 highlighted in Figure 31 have the objective of making an initial assessment of the environment. To prioritize which region to assess first, each mission was divided into four assessment tasks. The regions of the desired tasks for A1 and A2 are presented in Table 15, with the first tuple representing the initial xy point and the second the xy area size. Since the objective is to execute a fast visual evaluation of the environment, the sensors are not required and the assessment maneuvers fit better with the task, making it only achievable by UAVs.

Table 15 – Sequences of regions to be assessed by missions A1 and A2. Areas are delimited by the xy origin and xy size.

Task	A1	A2
a0	<2.0,2.0><33.0,10.0>	<35.0,2.0><33.0,10.0>
a1	<2.0,12.0><33.0,10.0>	<35.0,12.0><33.0,10.0>
a2	<2.0,22.0><33.0,10.0>	<2.0,22.0><33.0,10.0>
a3	<2.0,32.0><33.0,8.0>	<35.0,32.0><33.0,8.0>

Following the assessment missions, two more focused missions (B1 and B2) were required. These intend to make the robots travel around the buildings, followed by a search of the delimited area, as represented in Figure 32. To ensure such behavior, each mission is composed of 5 approach tasks with points P1, P2, P3, P4, P1 as their respective goals. In the sequence, robots must attempt a search in the delimited area.

After visiting the first four positions, we require again the goal P1 to ensure that the robot verifies all the building perimeter. The sequence points and search region are described in Table 16. The building 2 (to the right) was categorized as a factory, so we required that all tasks be done with the gas sensor ON, due to safety restrictions.

Figure 31 – The areas of interest for missions A1 and A2. Each mission is accomplished by exploring smaller areas one by one, from a0 to a3.

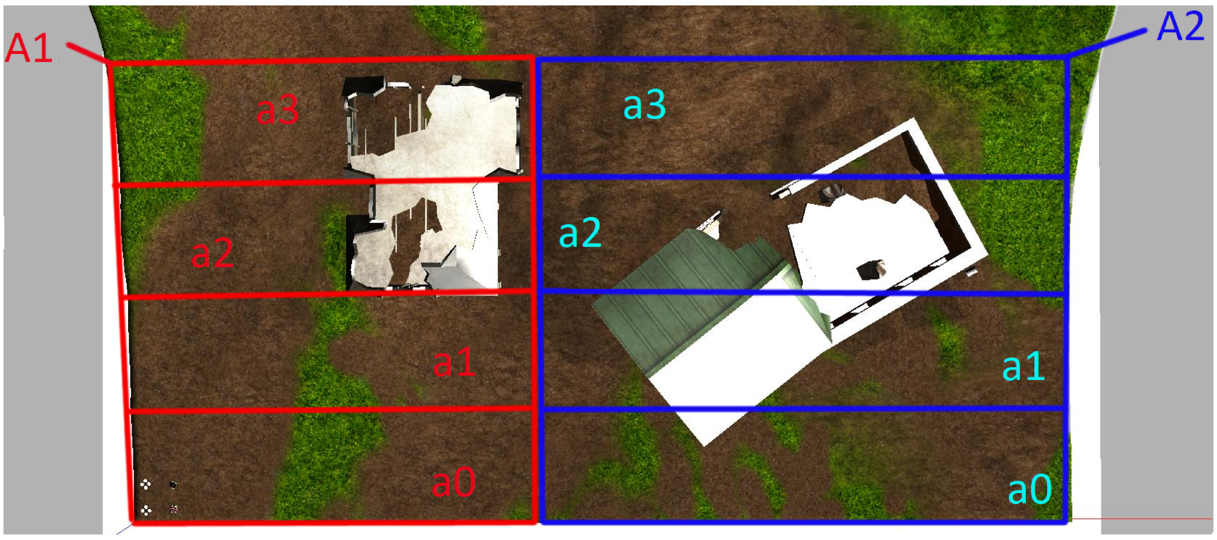


Table 16 – Sequences of tasks from missions B1 and B2.

Task	B1	B2
P1	<16.0,16.0,3.0>	<40.0,7.0,3.0>
P2	<32.0,16.0,3.0>	<64.0,20.0,3.0>
P3	<32.0,34.0,5.0>	<56.0,31.0,5.0>
P4	<16.0,34.0,5.0>	<32.0,18.0,4.0>
P1	<16.0,16.0,3.0>	<40.0,7.0,3.0>
Exp	<16.0,16.0><16.0,18.0>	<32.0,5.0><32.0,27.0>

As a result of initial discoveries made in the environment, by the suspicion of the presence of victims, the Task Force Leader might find it interesting to verify regions E1 (<10.0,20.0><5.0,10.0>) and E2 (<30.0,25.0><10.0,10.0>). These were represented by missions of only one task requiring the search of areas highlighted in Figure 33.

The described missions were inserted and sent to the Task Dispatcher in the sequence presented in Table 17. Mission RB_UGV2 represents the direct requisition for the return to base of robot UGV2. Each mission received a specific priority, with E1 and E2 the ones that should be more prioritized, since the lower the value, the greater the desire for its execution We have not waited for the conclusion of the assessment of the whole area to require the next missions. So we could reduce the simulated time and force all the robots to be assigned at the same time.

7.1 ROBOTS RESPONSE TO ENVIRONMENTAL CHANGES

All robots state changes were monitored, which resulted in the dynamics shown in Figure 34. The moments when tasks have been assigned are highlighted in the

Figure 32 – Delimitation of approach points and area of exploration to be executed by missions B1 and B2. The Xs over building 2 represent points with gas leakage.

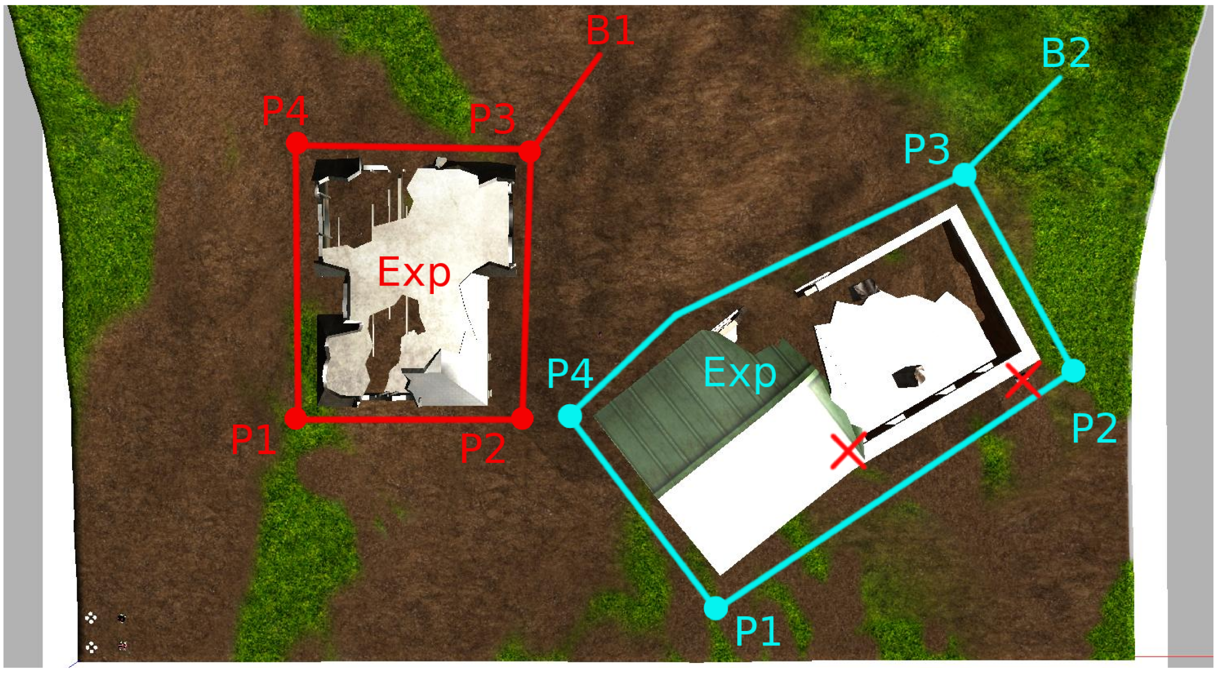
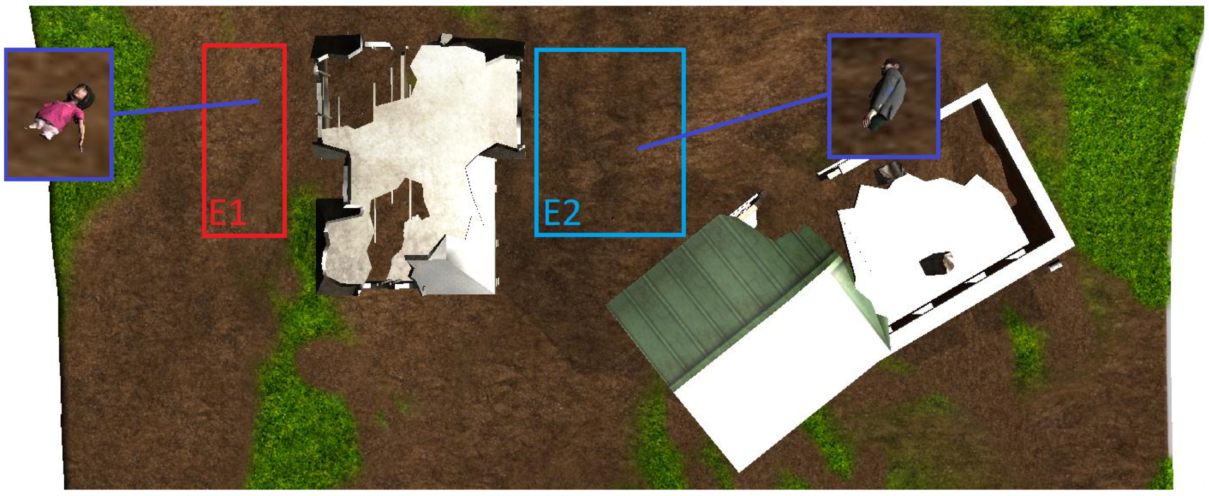


Figure 33 – External search missions E1 and E2. Both covered areas have the presence of a victim.

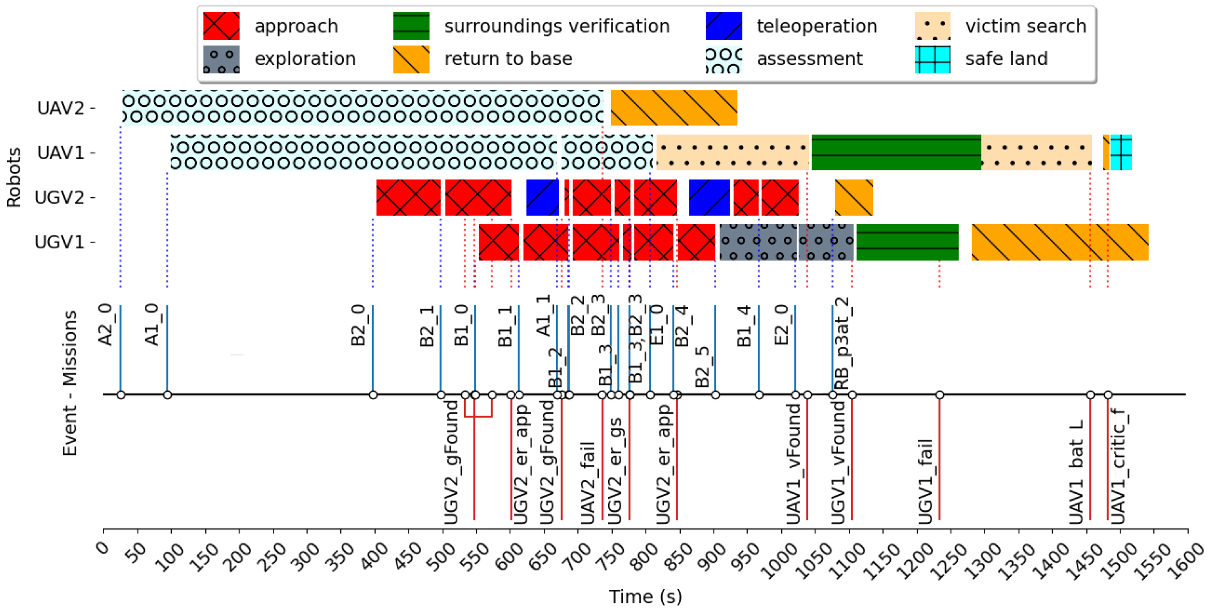


timeline by the vertical lines above the central horizontal line. And the vertical lines below the line represent events we considered important to demonstrate how the robots follow the modeled specifications. Above the timeline, we present the task being executed by each robot over time. For example, at 400 seconds, we can notice that task B2_0 (first task of mission B2) was assigned to the UGV2 since the dotted line ends on UGV2 bar graph.

Table 17 – Missions priorities and moment they were inserted in the HRI.

Time (s)	Mission	Priority
25	A2	5
94	A1	4
397	B2	3
549	B1	2
807	E1	0
1021	E2	0
1078	RB_UGV2	5

Figure 34 – Influence of events over maneuvers executed by robots. The bar graphs represent the maneuvers performed by each robot, and the timeline shows when tasks are assigned by TD or events occur in the robots.

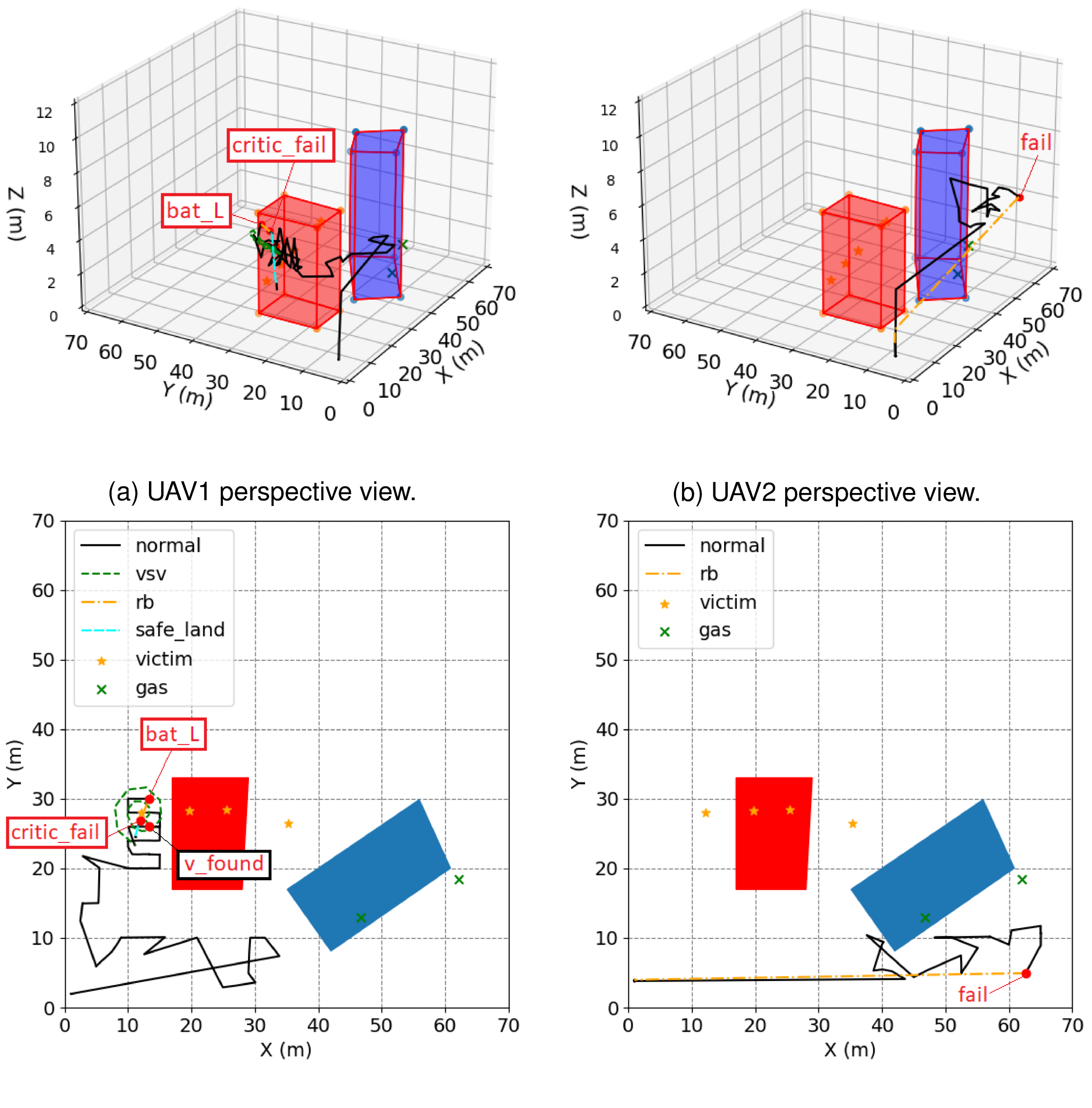


From the graph presented in Figure 34 it is possible to notice that robots quickly start the tasks assigned by the TD. It is also possible to evaluate the behaviors of robots in the face of unexpected situations. After about 740 seconds, a failure occurred on UAV2, which reacted by returning to base since it changed to Degraded Mode I. Figure 35 presents the UAVs trajectories, and the UAV2 correctly aborts the assessment of mission A1 and immediately returns to base. Therefore, avoiding increasing the risk of falling into the worksite, where it could harm a victim or a member of the USAR team.

UAV1 successfully finished the task A1_0 at moment 670 s and the TD immediately assigned to it the second task A1_1. The UAV1 assumes another task only after the Task Force Leader requires the mission E1 (807 s), leading the robot to its execution due to its position and A1 lower priority. This is an interesting point to notice. The TD makes the robot preempt the current task to execute something more important. A similar behavior could be expected after the UAV2 recognized a failure, though nothing happened because the other robots were already executing tasks with higher priorities.

In Figure 35a is shown that while executing the mission E1, the UAV1 recognized the presence of a victim at moment 1040 s. As a result, it changed to the VICTIM mode of operation, where the ‘victim surrounding verification’ maneuver is required. Figure 36 shows a closer view of the maneuver execution, represented by a spiral movement executed around the victim. After the execution of maneuver VSV, the robot resumed the execution of mission E1, continuing the zig-zag motion, till the UAV1 battery level changed to LOW, forcing the robot to return to base.

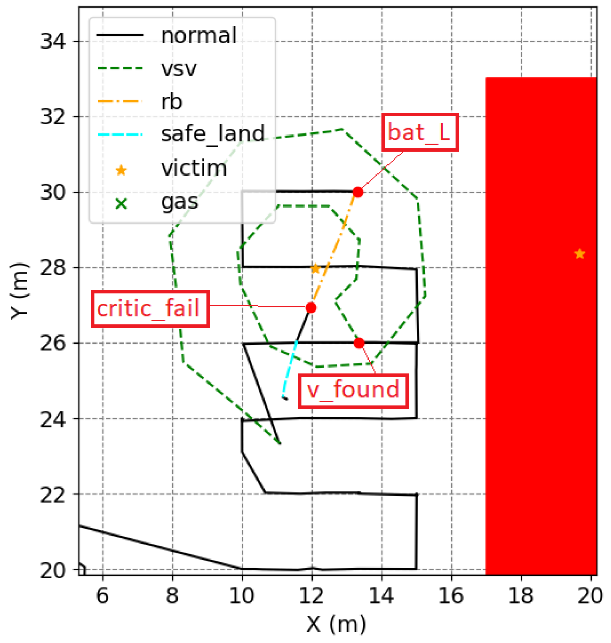
Figure 35 – Trajectories executed by the UAVs and their reaction to unexpected events. UAV2 returns to base after recognizing a failure and UAV1 verifies a victim surrounding, tries to return to base and land in the scene due to a critical failure. Red dots highlight events that affect the UAVs behaviors.



(a) UAV1 perspective view. (b) UAV2 perspective view.

(c) UAV1 top view. (d) UAV2 top view.

Figure 36 – UAV1 behavior in the face of unexpected events. The UAV1 changes its behavior after finding a victim so that it accomplishes the VSV maneuver. After the surroundings are verified, the mission E1 is reestablished till the events bat_L and critic_fail degrade the behavior of the UAV.



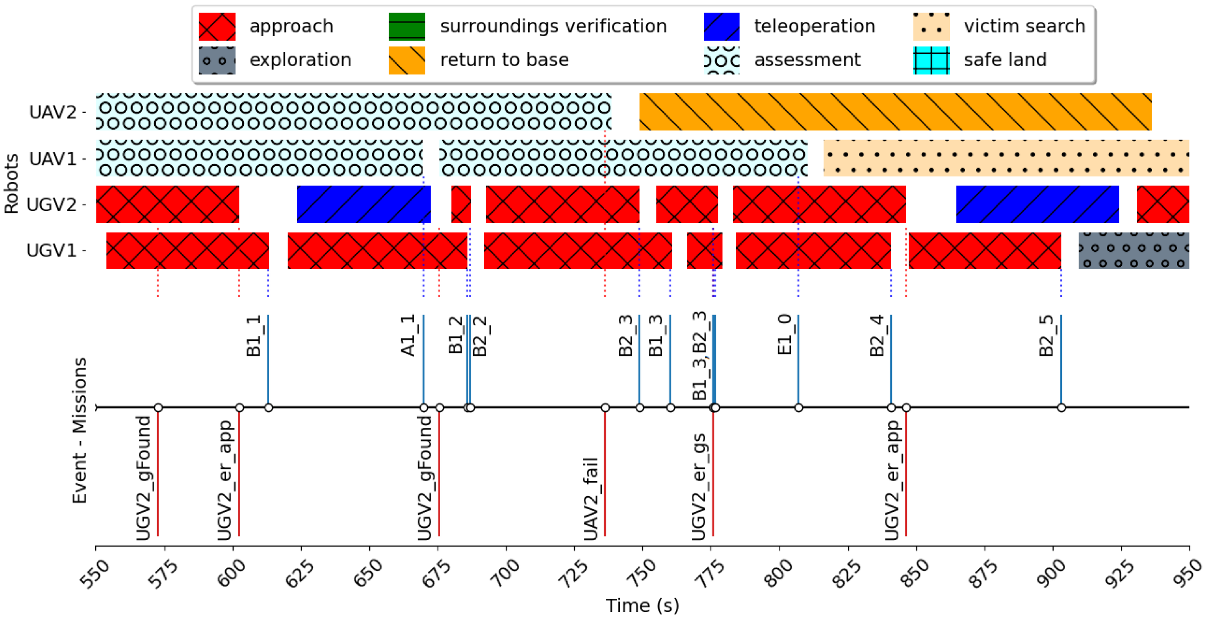
To verify the response of robots to a critical failure that occurred close to a victim, we forced such a failure while the UAV1 was passing above the victim. As expected, it changed to the CRITIC SYSTEM operation mode and executed the safe land maneuver. This is accomplished by landing the robot at a distance greater than 3 meters away from any victim, which was correctly executed as shown in Figure 36. Instead of just moving down, the UAV1 also moved away from the victim.

An interesting behavior of the UGVs can be better explained by focusing on events that occurred between 550 s and 950 s, highlighted in Figure 37. At 775 seconds, both robots were performing tasks B1_3 and B2_3 when the UGV2 gas sensor failed. As a result of the failure, both tasks were reassigned to maintain the execution of the missions, assigning the B2 to the robot that still has gas sensor working, since it is required by the mission. Figure 38 demonstrates the exact moment where the UGVs change position. UGV2 goes from building B2 to B1, and UGV1 goes from building B1 to B2.

An interesting behavior of the UGVs can be better explained by focusing on events occurred between moment 550 s and 950 s, highlighted in Figure 37. Notice that in moment 775 s both robots were executing tasks B1_3 and B2_3 when the UGV2 gas sensor stopped working. As a result of the failure, both tasks were reassigned to maintain the execution of the missions, assigning the B2 to the robot that still has gas sensor working, since it is required by the mission. Figure 38 demonstrate the exact moment where the UGVs change position, UGV2 goes from building B2 to B1 and UGV1 goes from building B1 to B2.

Such behavior demonstrates one of the advantages of having a system composed of a reactive layer tied to a deliberative planner. The reactivity of the system can not only ensure the safety and reliability of robots, but also increase their efficiency by triggering mission assignment replanning.

Figure 37 – Maneuvers executed by the robots from 550 to 950 s. The UGV2_er_gs occurred in 775 s immediately triggered the reallocation of robots to maintain the maximization of tasks being executed.

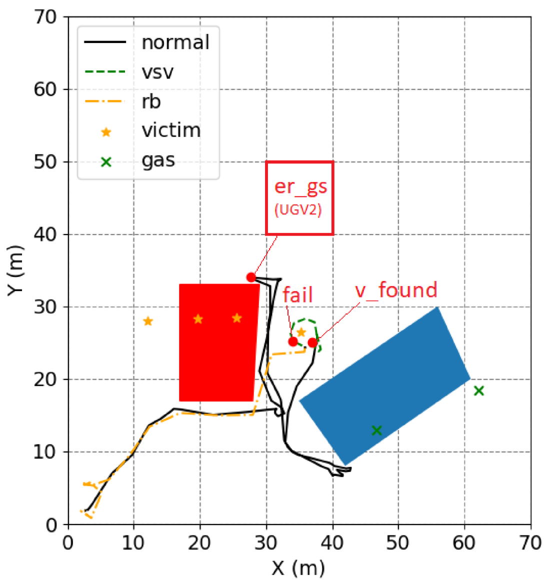


Another interesting situation to be noticed in figures 37 and 38 is the UGV2 teleoperation, which was required at two different moments. First, when the UGV2 gets too close to the gas source and can not find a way to transpose it. Second, when it cannot plan a path around the corner of building 1, which may have happened due to some errors in mapping. At both moments, the robot changed to HRI mode of operation and required human control because UGV2 was unable to plan a path by itself. It shows the benefit of the semi-autonomous approach allied to the SCT-based control system, that is, the reduction of the load on the human monitoring and controlling the robots. Since the robots can define the moment they need help, humans do not have to try paying attention to all robots at the same time.

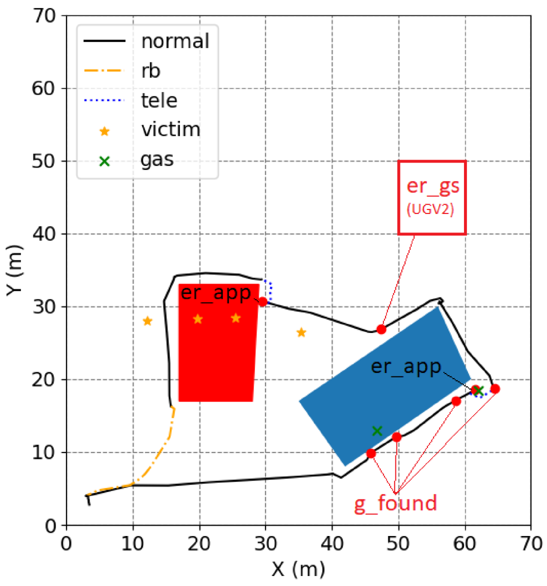
It is possible to notice in Figure 38b that in four moments, the robot reported the detection of gas leakage. The gas sensor was implemented in such a way that it generates a *gas_found* event only if the last reported point is farther than 4.5 m from the current position and it has gas recognized. In this work we have not assigned a special behavior to robots that are too close to gas sources, as it was made for the presence of victims. Since the situation faced by the UGV2 could lead to explosions, it would be interesting to model levels of danger that could prevent the robot from getting too close to the source.

After the mission E2 was required, the UGV1 aborted the last task of B2 and looked for victims in the assigned area. When a victim was found it executed the VSV maneuver, which for the UGVs is characterized by simply moving around the victim in a circle with both sensors ON. Observe in figures 34 and 38 that the robot faces a failure while executing the surrounding verification, but in a different approach compared to

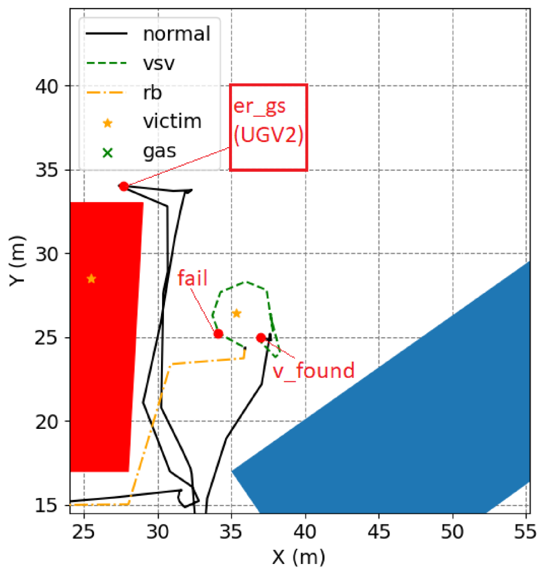
Figure 38 – Trajectories executed by the UGVs. Observe how both robots change their trajectories due to the failure of the gas sensor on UGV2.



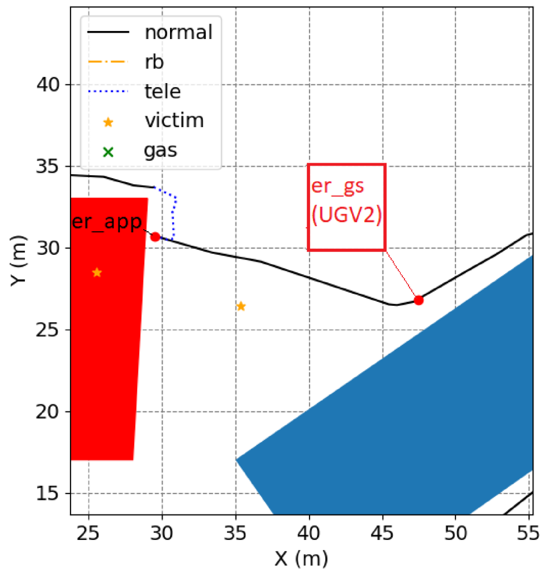
(a) UGV1.



(b) UGV2.



(c) UGV1 zoom.



(d) UGV2 zoom.

the UAV, UGVs can finish the current task before returning to base.

7.2 MCS COMPUTATIONAL COST

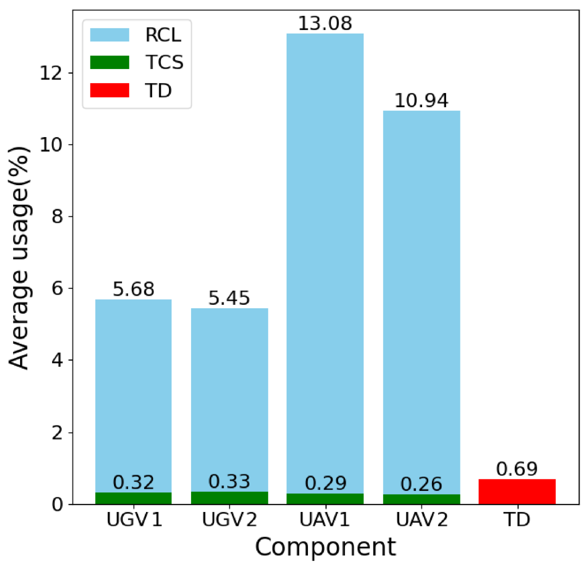
Besides analyzing the behavior of the robots in different unexpected situations, it is fundamental to evaluate the feasibility of implementing the MCS architecture in real systems. The robots implemented in USAR are generally embedded with systems that require high computational cost, e.g., localisation, mapping, and path planning algorithms. So, less effort required for the TCS means more for these low-level systems. In this Section, we compare the cost of implementing all the low-level control systems to the requirements imposed by the TCS.

We have evaluated the Memory and CPU usage of the implemented system. All the simulations were implemented in a Virtual Machine (VM) provided by the Google Cloud Platform (GCP). The implemented VM was a c2-standard-8 composed by a Cascade Lake processor with 8 cores and 32GB of RAM memory, and Ubuntu 18.04 as the operational system.

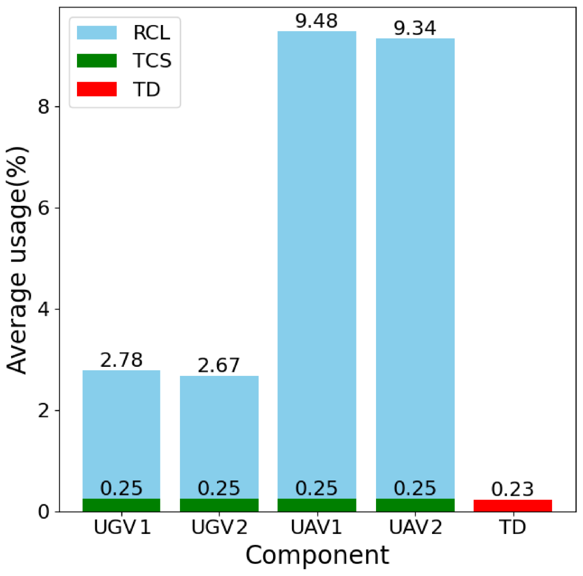
In Figure 39, we show the VM average percentage of CPU and RAM usage related to the TCS and RCL layers of each robot, as well as the TD usage. While the TCS layer of robots consumed an insignificant amount of CPU (about 0.3%), the Robot Control Level required more effort (up to 5.68% for UGVs and 13.08% for UAVs). If we verify the average usage of the embedded system, the TCS required 5.7% of the CPU of UGV2 and only 2.3% of the CPU of UAV2. Remaining more than 94% of the CPU for the RCL. The same low influence is verified in the memory consumption, which makes us believe that the implementation of the TCS in the robots is not a limitation for the scalability of robots.

It was not possible to evaluate the system with more than four robots due to the computational cost of the simulation. So we cannot determine the correct behavior of the TD in the presence of more robots on the system. In the simulated scenario, it consumed only 0.69% of the CPU and 0.23% of the processing memory. However, the inclusion of robots could cause it to grow exponentially because the A* search algorithm has a computational complexity of $O(b^d)$, with b representing the ramification factor (quantity of tasks with the highest priority) and d the depth of the goal state (smallest quantity among tasks and robots). Simulations with more robots and different combinations of missions are required to have an accurate understanding of the TD scalability, since the implemented algorithm preempts when all robots or tasks are assigned, which may reduce the cost.

Figure 39 – Computational cost of the proposed architecture. The implementation of the TCS and TD has a very low impact on the computational cost when compared to the effort required by the RCL.



(a) CPU usage.



(b) MEMORY usage.

8 CONCLUSIONS

Post-disaster scenes are very unpredictable, dangerous, and unstructured. What makes the use of robots very attractive is their ability to assess subhuman locations and regions with possible presence of hazardous materials, besides bringing the possibility of an efficiency increase in the victims' search. However, their actual implementation in field situations still necessitates improvements not only in terms of action efficiency but also in terms of behavior robustness.

The lack of safety and reliability in robotic systems applied to USAR highlighted by Delmerico et al. (2019) has motivated the development of the Multi-robots Coordination System proposed in this work. Besides, the deliberative-reactive architecture proposed by Battistella (2015) demonstrated to be an interesting approach to increase the system's reliability while maintaining the execution of efficient actions. But while Battistella (2015) focused on assigning multiple tasks to one robot and defining the best sequence to execute them, the approach presented here coordinates the assignment of tasks to multiple robots which, by their internal control system, ensure the correct behavior to attain the modeled specifications.

As far as we know, this work has presented the first implementation of formal models based on the SCT applied to the control of robots mutually executing tasks in USAR. Here we developed the models based on the bibliographic overview and procedures usually followed by USAR teams. The modeled behaviors and payloads presented here are just one among many possible approaches. As mentioned in Section 5.4, here we attempted to increase the endurance, safety, and reliability of the robots, and we have accomplished it as shown by the study cases from Chapter 7. By the degraded modes of operation, the robots become more robust to failures (the robot changes its behavior if needed to ensure safety) and an endurance increase is noticed, since robots can be used even with a certain level of degradation.

A control system based on the Supervisory Control Theory has the advantage of being mathematically proved to be correct, thus being able to guarantee more reliability and predictability to the system where it is applied. By implementing the supervisors synthesized in this work, we guarantee that the required specifications are met, thus increasing the safety of the actions executed by the robots in such an unstructured environment as an USAR disaster. The study case presented in Chapter 7 showed the correct reaction of the robots to unexpected events, where they changed the behavior to overcome undesired situations, like when the UAV faces a critical failure and immediately reacts executing the safe land. It also shows the synergy between the reactive and deliberative layers, that by their interactions ensure the best application of the robots, e.g., when a failure on the gas sensor of one UGV triggers the reallocation of both UGVs tasks.

For the correct implementation of the SCT it is fundamental that the robot monitor all the events executed or recognised by the sensors so that the current state can be defined. Therefore, by saving all the changes in the robot state, we can grant the capability of verifying all the steps executed by the robot. Also, all the communication between robots and the TD have been stored, making possible further analysis of the employed strategy. With these information, the developed system follows the ethical principles of Design (2019), by granting transparency and ensuring the accountability of designers and operators.

Since the maneuvers models were implemented at a higher abstraction level, their implementation is still not too restrictive, making it possible to implement any possible approach to control the specific motion patterns. As a result, the developed approach is not limited to the packages here implemented, and because the full implementation is available at SIMON (2021b), we strongly recommend that new developed maneuvers procedures be implemented with the MCS as the base architecture. Other works that would benefit from the approach presented here include those aiming to optimize the tasks assigned to robots in USAR.

According Murphy et al. (2008), the simplicity of use is also a very important point to be attained for the actual deployment of a new technology applied to USAR. We demonstrated a system that, as a result of the reactive-deliberative allocation system and the constant feedback of the robots, tends to reduce the load on the human interacting with the HRI. Also, the way that the supervisors have been implemented abstracts the intrinsic complexity of controllers based on SCT, making it almost invisible to the human user.

8.1 FUTURE WORKS

The development of this work and the results obtained from the simulated environment were demonstrated to be very relevant and could be considered as a step forward in the actual implementation of more autonomous robots in the USAR field. However, it did raise unresolved issues that could be addressed in future works. A possible improvement would be changing the behavior of the gas sensor by adding levels of danger to it that could lead to different sets of enabled maneuvers, avoiding robots getting too close to places with high chances of explosion.

Also, the following two steps are highly recommended for increasing the architecture acceptance. First, we recommend that it be implemented with a larger number of robots in order to demonstrate the system's scalability. Secondly, the development of a real test scenario with the implementation of this architecture in robots composed by the USAR latest robotic technologies.

A very important issue present in USAR robotics, highlighted by Murphy et al. (2008) and approached by Vittorio Amos Ziparo et al. (2007) and Kleiner and Dornhege

(2007), is the lack of constant communication. Since in this work we implement the supervisors directly on each robot, they are still efficient in the case of communication failures between robots. But in such situations, the allocation of tasks becomes unfeasible, which leads the robots to idle when finishing the already allocated tasks. Considering the communication failures directly in the models could increase the robots' reactivity to the environment. But a better improvement would be to upgrade the architecture to a non-centralized one. For example, the evolution of the architecture to one based on multi-agents or swarms could be a possible way of increasing the robots' deliberation capability.

A last improvement would be the development of a mechanism that makes it possible to consider robots interactions in the formal models applied to the MCS. For example, make one robot's internal behavior dependent on events recognized by others.

BIBLIOGRAPHY

ARNOLD, Ross D; YAMAGUCHI, Hiroyuki; TANAKA, Toshiyuki. Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. **Journal of International Humanitarian Action**, Springer, v. 3, n. 1, p. 18, 2018.

BATTISTELLA, Sandro. **Controle de missão baseado na teoria de controle supervisório com aplicação a veículos subaquáticos autônomos**. 2015. PhD thesis – UFSC - Universidade Federal de Santa Catarina, https://repositorio.ufsc.br/xmlui/handle/123456789/158790.

BATTISTELLA, Sandro; QUEIROZ, Max H. de. Simulation environment of an architecture for mission control system of AUVs operating in lakes of hydroelectric dams. In: PROCEEDINGS of the 2014 3rd International Conference on Applied Robotics for the Power Industry. [S.l.: s.n.], 2014. P. 1–6.

CARBONE, Andrea; FINZI, Alberto; ORLANDINI, Andrea; PIRRI, Fiora. Model-based control architecture for attentive robots in rescue scenarios. **Autonomous Robots**, Springer, v. 24, n. 1, p. 87–120, 2008.

CASPER, Jennifer; MURPHY, Robin R. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 33, n. 3, p. 367–385, 2003.

CASSANDRAS, Christos G; LAFORTUNE, Stephane. **Introduction to discrete event systems**. [S.l.]: Springer Science & Business Media, 2009.

CNN. **September 11 Terror Attacks Fast Facts**. 2019. Available from: `https://edition.cnn.com/2013/07/27/us/september-11-anniversary-fast-facts/index.html`. Visited on: 9 Jan. 2020.

COSTELHA, Hugo; LIMA, Pedro. Robot task plan representation by Petri nets: modelling, identification, analysis and execution. **Autonomous Robots**, Springer, v. 33, n. 4, p. 337–360, 2012.

DAI, Xuefeng; JIANG, Laihao; LI, Dahui. Integrating predicate reasoning and reactive behaviors for coordination of multi-robot systems. **Cluster Computing**, Springer, v. 22, n. 3, p. 7413–7421, 2019.

DAI, Xuefeng; JIANG, Laihao; ZHAO, Yan. Cooperative exploration based on supervisory control of multi-robot systems. **Applied Intelligence**, Springer, v. 45, n. 1, p. 18–29, 2016.

DELMERICO, Jeffrey et al. The current state and future outlook of rescue robotics. **Journal of Field Robotics**, Wiley Online Library, v. 36, n. 7, p. 1171–1191, 2019.

DESIGN, Ethically Aligned. A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems. **Report (The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems, 2018)**, 2019.

DOROODGAR, Barzin; LIU, Yugang; NEJAT, Goldie. A learning-based semi-autonomous controller for robotic exploration of unknown disaster scenes while searching for victims. **IEEE Transactions on Cybernetics**, IEEE, v. 44, n. 12, p. 2719–2732, 2014.

EBADI, Toktam; PURVIS, Maryam; PURVIS, Martin. A framework for facilitating cooperation in multi-agent systems. **The Journal of Supercomputing**, Springer, v. 51, n. 3, p. 393–417, 2010.

FARINELLI, Alessandro; RAEISSI, Masoume M; BROOKS, Nathan; SCERRI, Paul, et al. Interacting with team oriented plans in multi-robot systems. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 31, n. 2, p. 332–361, 2017.

INSARAG. **INSARAG Guidelines**. INSARAG. Feb. 2015. Available from: `https://www.insarag.org/methodology/insarag-guidelines/`. Visited on: 30 May 2020.

KLEINER, Alexander; DORNHEGE, Christian. Real-time localization and elevation mapping within urban search and rescue scenarios. **Journal of Field Robotics**, Wiley Online Library, v. 24, n. 8-9, p. 723–745, 2007.

KLOETZER, Marius; MAHULEA, Cristian. Path planning for robotic teams based on LTL specifications and Petri net models. **Discrete Event Dynamic Systems**, Springer, p. 1–25, 2019.

KOO, T John; LI, Rongqing; QUOTTRUP, Michael M; CLIFTON, Charles A; IZADI-ZAMANABADI, Roozbeh; BAK, Thomas. A framework for multi-robot motion planning from temporal logic specifications. **Science China Information Sciences**, Springer, v. 55, n. 7, p. 1675–1692, 2012.

LIMA, Carolina de. **Controle Supervisório de um Sistema de Patrulhamento Multirrobôs com Arquitetura Deliberativa/Reativa**. 2019. MA thesis – UFSC - Universidade Federal de Santa Catarina.

LIU, Yugang; NEJAT, Goldie. Multirobot cooperative learning for semiautonomous control in urban search and rescue applications. **Journal of Field Robotics**, Wiley Online Library, v. 33, n. 4, p. 512–536, 2016.

LIU, Yugang; NEJAT, Goldie. Robotic Urban Search and Rescue: A Survey from the Control Perspective. **Journal of Intelligent & Robotic Systems**, v. 72, n. 2, p. 147–165, 2013. ISSN 1573-0409.

LOPES, Yuri K; TRENKWALDER, Stefan M; LEAL, André B; DODD, Tony J; GROSS, Roderich. Supervisory control theory applied to swarm robotics. **Swarm Intelligence**, Springer, v. 10, n. 1, p. 65–97, 2016.

MALIK, Robi; ÅKESSON, Knut; FLORDAL, Hugo; FABIAN, Martin. Supremica–an efficient tool for large-scale discrete event systems. **IFAC-PapersOnLine**, Elsevier, v. 50, n. 1, p. 5794–5799, 2017.

MURPHY, Robin R. **Disaster robotics**. [S.l.]: MIT press, 2014.

MURPHY, Robin R. Human-robot interaction in rescue robotics. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, IEEE, v. 34, n. 2, p. 138–153, 2004.

MURPHY, Robin R. Trial by fire [rescue robots]. **IEEE Robotics Automation Magazine**, v. 11, n. 3, p. 50–61, 2004.

MURPHY, Robin R.; TADOKORO, Satoshi; NARDI, Daniele; JACOFF, Adam; FIORINI, Paolo; CHOSET, Howie; ERKMEN, Aydan M. Search and rescue robotics. **Springer handbook of robotics**, Springer, p. 1151–1173, 2008.

NATH, Amar; ARUN, AR; NIYOGI, Rajdeep. A distributed approach for road clearance with multi-robot in urban search and rescue environment. **International Journal of Intelligent Robotics and Applications**, Springer, v. 3, n. 4, p. 392–406, 2019.

QUEIROZ, Max H. de; CURY, José ER. Controle supervisório modular de sistemas de manufatura. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, SciELO Brasil, v. 13, n. 2, p. 123–133, 2002.

QUEIROZ, Max H. de; CURY, José ER. Modular supervisory control of large scale discrete event systems. In: DISCRETE Event Systems. [S.l.]: Springer, 2000. P. 103–110.

QUEIROZ, Max H. de; CURY, José ER. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In: IEEE. SIXTH International Workshop on Discrete Event Systems, 2002. Proceedings. [S.l.: s.n.], 2002. P. 377–382.

RAMADGE, P. J.; WONHAM, W. M. Modular supervisory control of discrete event systems. **Mathematics of Control Signals and Systems**, v. 1, n. 1, p. 13–30, 1988.

RAMADGE, P. J.; WONHAM, W. M. Supervisory control of a class of discrete event processes. **SIAM journal on control and optimization**, SIAM, v. 25, n. 1, p. 206–230, 1987.

REPUBLIC OF HAITI, Government of the. **Action Plan for National Recovery and Development for Haiti**. 2010. Available from:

`https://www.humanitarianlibrary.org/resource/action-plan-national-recovery-and-development-haiti-2`. Visited on: 3 Mar. 2020.

ROBI MALIK. **Supremica IDE**. [S.l.: s.n.], 15 Feb. 2021. Available from: `https://www.cs.waikato.ac.nz/~robi/download_waters/`.

SILVA CARDOSO, Priscila da; VIEIRA, Rosemary. O Megadesastre de Janeiro de 2011 na cidade de Nova Friburgo, Rio de Janeiro: aspectos históricos desde a colonização suíça e as condicionantes físicas. **Investigaciones Geográficas**, n. 52, p. 47–70, 2016. ISSN 0719-5370.

SIMON, M. E. **TCS code generator**. 2021. Available from: `https://github.com/mSimon12/TCS_supervisor`. Visited on: 29 Oct. 2021.

SIMON, M. E. **USAR multi-robots**. 2021. Available from: `https://github.com/mSimon12/usar_multirobot`. Visited on: 29 Oct. 2021.

STATHEROPOULOS, M; AGAPIOU, Agapios; PALLIS, George C; MIKEDI, Katerina; KARMA, Sofia; VAMVAKARI, J; DANDOULAKI, Miranda; ANDRITSOS, Fivos; THOMAS, CL Paul. Factors that affect rescue time in urban search and rescue (USAR) operations. **Natural Hazards**, Springer, v. 75, n. 1, p. 57–69, 2015.

TALAMADUPULA, Kartik; BENTON, J; KAMBHAMPATI, Subbarao; SCHERMERHORN, Paul; SCHEUTZ, Matthias. Planning for human-robot teaming in open worlds. **ACM Transactions on Intelligent Systems and Technology (TIST)**, ACM New York, NY, USA, v. 1, n. 2, p. 1–24, 2010.

WILLIAMS, Adam; SEBASTIAN, Bijo; BEN-TZVI, Pinhas. Review and Analysis of Search, Extraction, Evacuation, and Medical Field Treatment Robots. **Journal of Intelligent & Robotic Systems**, Springer, p. 1–18, 2019.

ZIPARO, Vittorio A; IOCCHI, Luca; LIMA, Pedro U; NARDI, Daniele; PALAMARA, Pier Francesco. Petri net plans. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 23, n. 3, p. 344–383, 2011.

ZIPARO, Vittorio Amos; KLEINER, Alexander; FARINELLI, Alessandro; MARCHETTI, Luca; NARDI, Daniele. Cooperative exploration for USAR robots with indirect communication. **IFAC Proceedings Volumes**, Elsevier, v. 40, n. 15, p. 554–559, 2007.

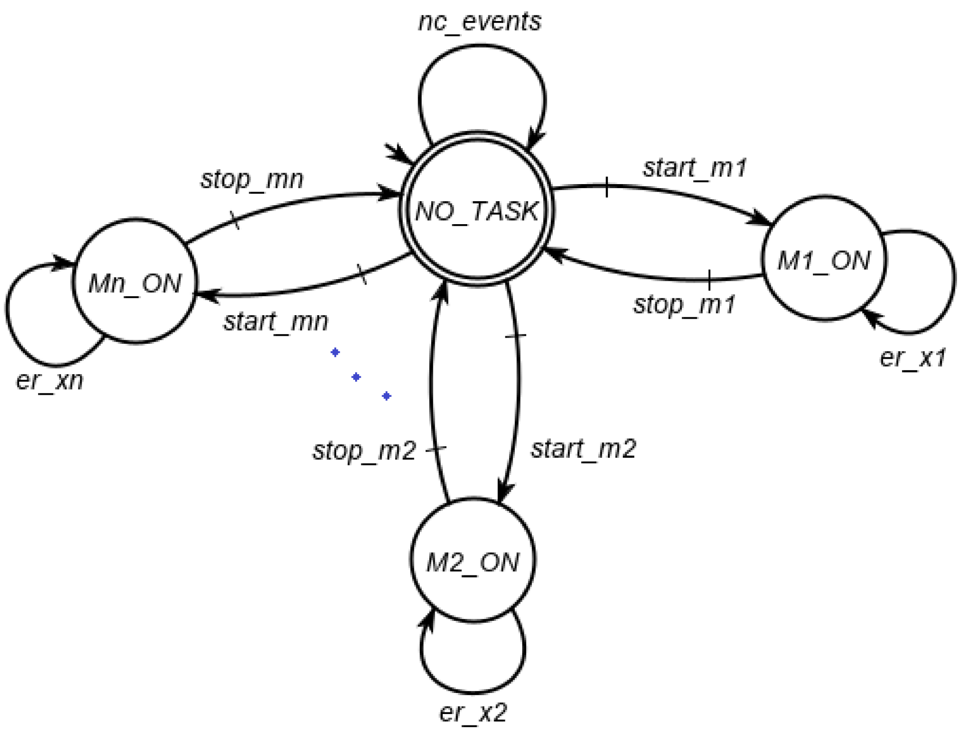
APPENDIX A – SPECIFICATIONS FULL DESCRIPTION

The correct functioning of the system, in order to guarantee the safety of the humans involved and of the robot itself, requires the control of the interaction between the components modeled on Section 5.1. As presented earlier, to guarantee the specifications classes highlighted on Section 5.4, we have synthesised 14 supervisors for the UGV and 11 for the UAV. Below we describe each one of the specifications that resulted on the implemented supervisors.

A.1 MANEUVERS MUTUAL EXCLUSION

All maneuvers make use of the robot's locomotion system, making their concurrent execution impossible. However, only the plant models do not prevent the maneuvers mutual activation, so it is necessary to guarantee mutual exclusion through a specification. This is accomplished by the model in Figure 40, in which the beginning of a maneuver is linked to the end of another one that is in execution.

Figure 40 – Maneuvers Mutual Exclusion.



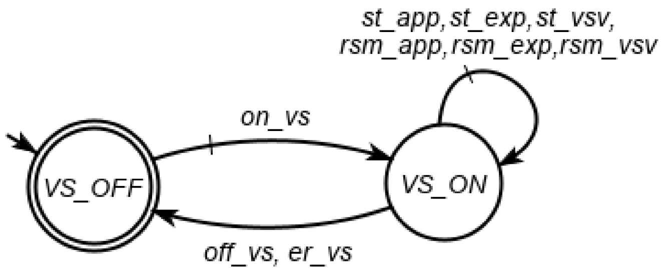
For better understanding, we have abstracted the events applied by the described below. Observe that since the start events are not present on the peripheral nodes, they are not enabled, so avoiding them the mutual execution between maneuvers.

- *nc_events* = all non-controllable events present from the automaton alphabet;
- *start_mi* = all events that make maneuver *i* change to execution mode;
- *stop_mi* = all events that make maneuver *i* change stop execution mode;
- *er_xi* = event error related to all except *i* maneuver.

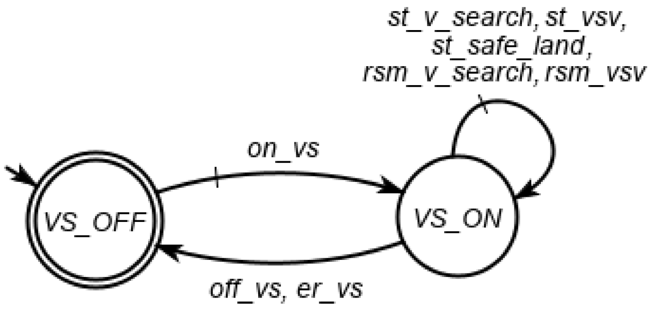
A.2 MANEUVERS DEPENDENCIES ON SENSORS

The execution of a maneuver depends on the activation of the sensors that it makes use. Therefore, we restricted the start of maneuvers to the previous activation of the used sensors, see Figure 41 and Figure 42, for the UGV. Note that the return to base maneuver is not dependent on sensors, as it must perform movements in known regions. In addition, we limited the UGV execution of VSV and EXP maneuvers to the use of both sensors, since they are tasks aimed at maximizing the information provided to the Task Force Leader and generally performed in conditions where there is almost none initial information about the scene.

Figure 41 – Maneuvers Dependencies on Victim Sensor.



(a) UGV



(b) UAV

A.3 SENSORS TURN OFF CONDITION

As opposed to the previous specification, we must ensure that the sensors are not turned off when they are in use by any of the maneuvers. For this, the models in

Figure 42 – Maneuvers Dependencies on Gas Sensor.

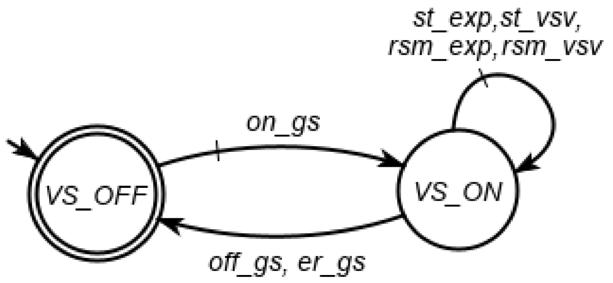
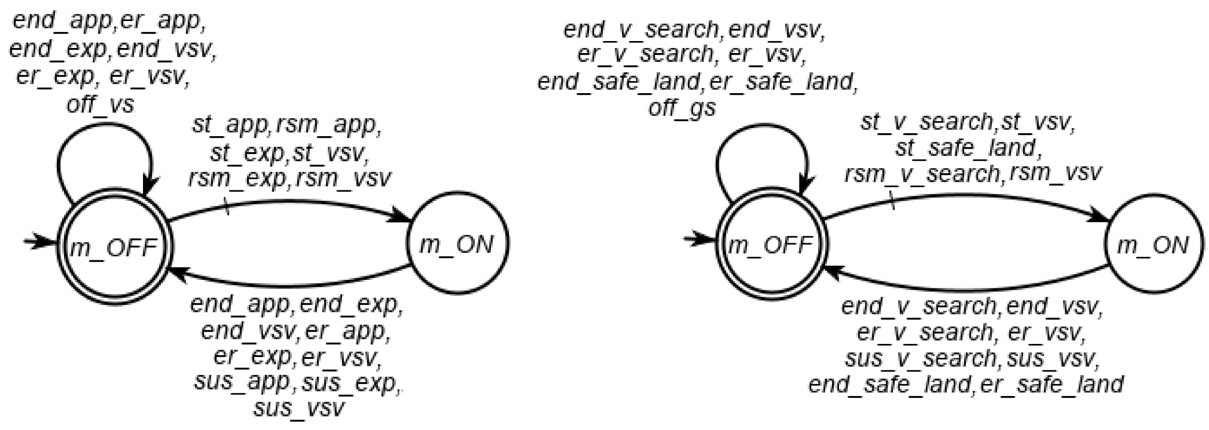


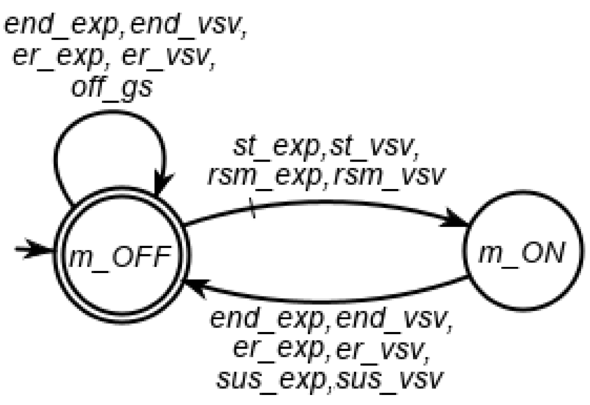
Figure 43 and Figure 44 inhibit the events off_vs and off_gs while some dependent maneuver is in execution.

Figure 43 – Victim Sensor Turn Off Condition.



(a) UGV (b) UAV

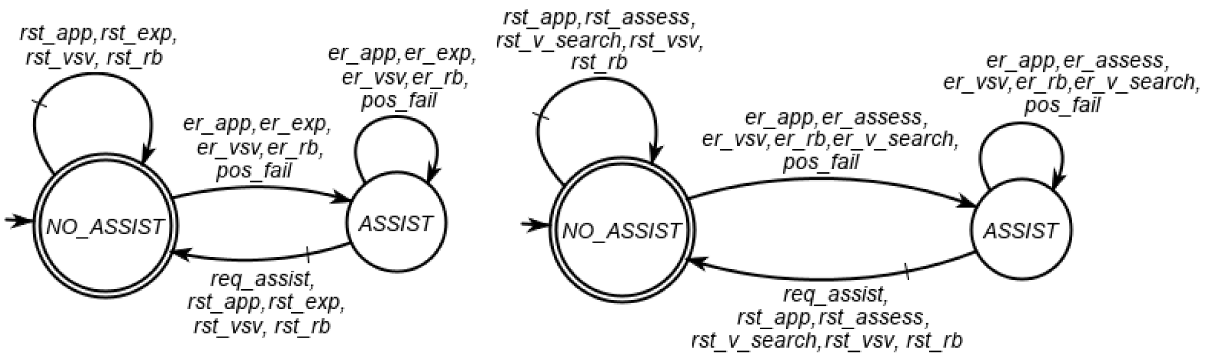
Figure 44 – Gas Sensor Turn Off Condition.



A.4 ASSISTANCE REQUESTS

As mentioned before, it is possible that due to some unexpected situation the robot becomes unable to finish the maneuver being executed. Which triggers an error event represented by er_mi, where *mi* is the task being executed. In such situations, we have considered that the robot must request human assistance, in a semi-autonomous approach like presented by Doroodgar, Liu, and Nejat (2014). In order to ensure that assistance requests are made only when necessary, avoiding unnecessary deviations from the attention of human teams, we propose, through the model in Figure 45, to enable the request only after the occurrence of a maneuver failure or if the robot faces a failure on the positioning sensors (*pos_fail*).

Figure 45 – Assistance Requests.



(a) UGV

(b) UAV

A.5 TELEOPERATION PRIORITY

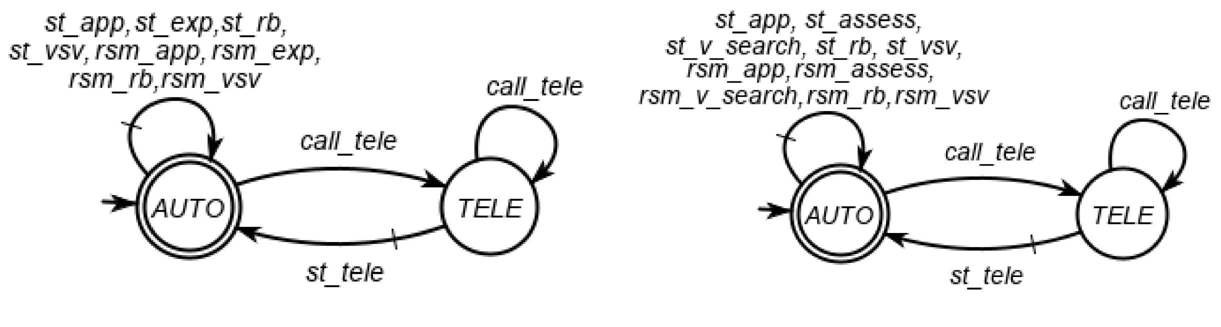
The teleoperation of robots is mainly applied in two situations: 1) when the robot requires assistance, like previous specification; or 2) due to the human intention to directly control the robot. Both cases require a human requesting teleoperation access. So, to ensure this behavior, we use the specifications in Figure 46 to enable the st_tele event only when after the human control intention is confirmed.

In order to avoid the system delay in granting access to the teleoperated mode, as it is performing other maneuvers, the autonomous maneuvers are disabled when on TELE state. This makes the system prioritize the activation of the teleoperated mode, preventing the person responsible for the control from waiting indefinitely for the robot's control to be released.

A.6 VICTIM SURROUNDINGS VERIFICATION START CONDITION

Since the purpose of the VSV maneuver is to check the victim's surroundings, we have as a precondition for its execution the need to locate the victim. Therefore,

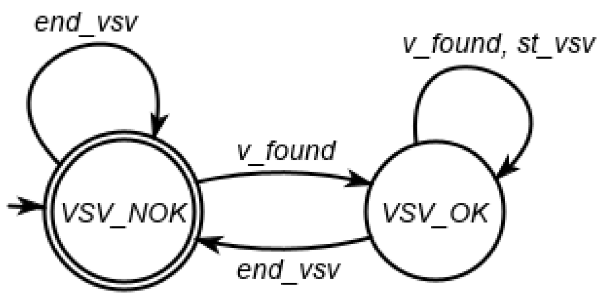
Figure 46 – Teleoperation Priority.


st_app, st_exp, st_rb, st_vsv, rsm_app, rsm_exp, rsm_rb, rsm_vsv
call_tele
call_tele
AUTO
TELE
st_tele
st_app, st_assess, st_v_search, st_rb, st_vsv, rsm_app, rsm_assess, rsm_v_search, rsm_rb, rsm_vsv
call_tele
call_tele
AUTO
TELE
st_tele

(a) UGV (b) UAV

to prevent the system from executing at an inappropriate moment, we propose the specification of Figure 47 which conditions the activation of the VSV maneuver to the previously identification of a victim.

Figure 47 – Victim Surroundings Verification Start Condition.



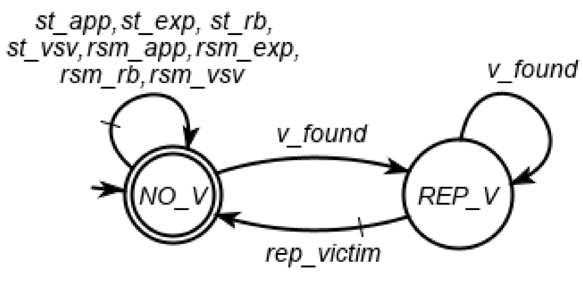
A.7 VICTIMS AND GAS REPORT CONDITION

We need to ensure that when identifying a victim or a gas source, the robot informs the Task Force Leader as soon as passible about this finding. For this end, through the models in Figure 48 and in Figure 49, we disable the execution of robot movements while a finding (victim or gas) is not reported. What forces the robot to brief the Task Force Leader before resuming the task.

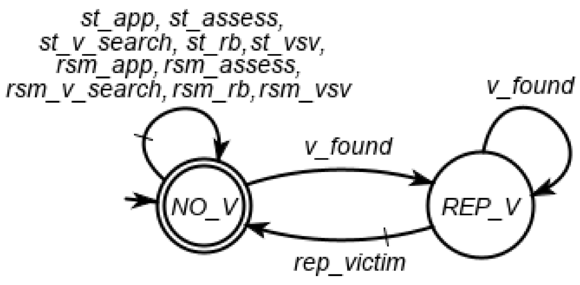
A.8 SIMPLE FAILURE OPERATION MODE

Currently, there are robotic systems capable of adapting to simple failures, thus allowing the completion of tasks with a degraded behavior, e. g. perform safe movements even after failure of one of four engines. For this type of failure we disable some maneuvers, but we still enable the return to base for repairs, with no total blocking of the maneuvers, see Figure 50. Here if a robot is executing a maneuver when a simple

Figure 48 – Victim Report.

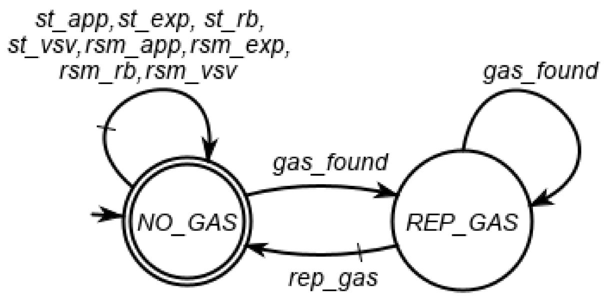


(a) UGV



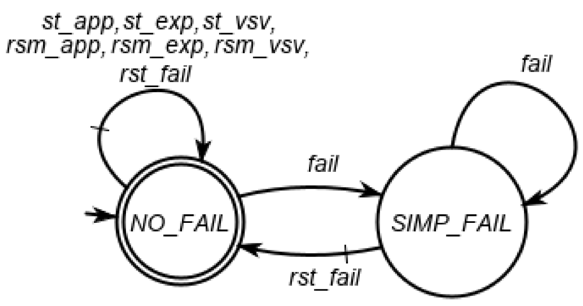
(b) UAV

Figure 49 – Gas Report.

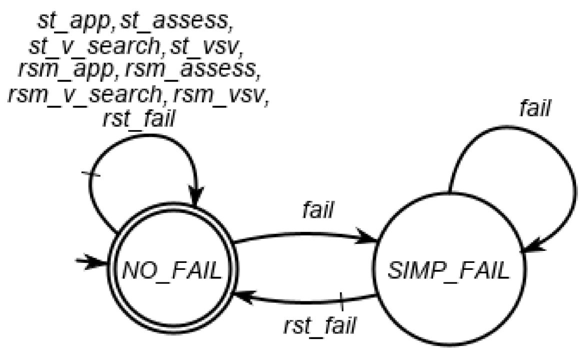


failure occurs, the TM is responsible for defining if the robot must first end the task or if it should go to base immediately.

Figure 50 – Simple Failure Operation Mode.



(a) UGV

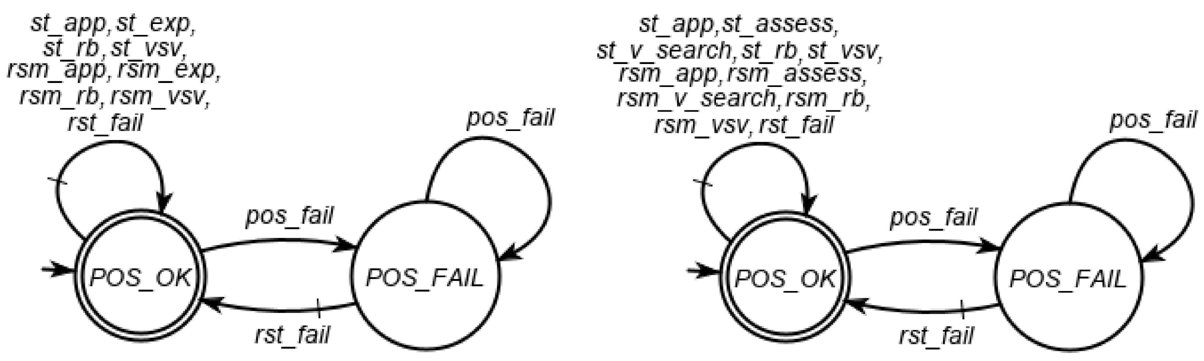


(b) UAV

A.9 POSITION FAILURE OPERATION MODE

To ensure the safety of people close to the robot, whether victim or rescuers, it is necessary to prevent the robot from performing uncertain movements. The occurrence of the pos_fail event represents some failure of the robot's localization system, making it incapable of performing autonomous movements in a safe way. Therefore, by the specification presented in Figure 51, we blocked the execution of maneuvers and movements after the position failure event occurred, enabling only teleoperation, and the safe land in UAV case.

Figure 51 – Position Failure Operation Mode.

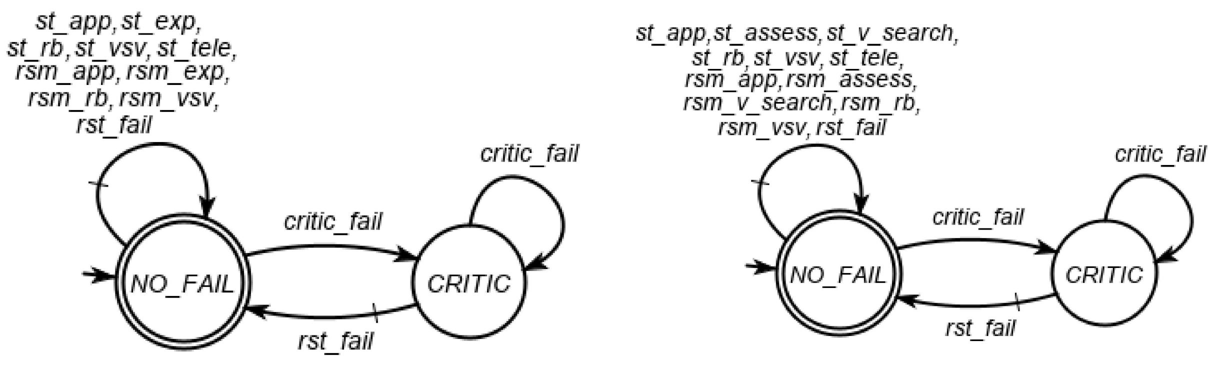

st_app, st_exp, st_rb, st_vsv, rsm_app, rsm_exp, rsm_rb, rsm_vsv, rst_fail
pos_fail
pos_fail
POS_OK
POS_FAIL
rst_fail
st_app, st_assess, st_v_search, st_rb, st_vsv, rsm_app, rsm_assess, rsm_v_search, rsm_rb, rsm_vsv, rst_fail
pos_fail
pos_fail
POS_OK
POS_FAIL
rst_fail

(a) UGV (b) UAV

A.10 CRITIC FAILURE OPERATION MODE

If the failure is more critical, e. g. crucial components in malfunction, it becomes impossible to perform any autonomous maneuver, as well as teleoperated control. Therefore, to ensure people's safety, the UGV system is completely blocked by the specification presented in Figure 52a. In this condition, the robot's only possibility would be to inform its position for future rescue and repair, as well as in the case of a critical battery. And since the UAV would be away from the ground with possible object or people below it, the system maintains safe land still enabled, as showed on Figure 52b.

Figure 52 – Critic Failure Operation Mode.

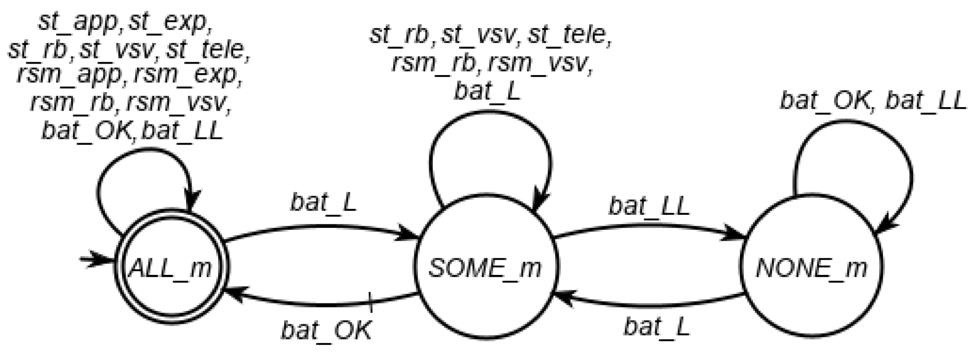

st_app, st_exp, st_rb, st_vsv, st_tele, rsm_app, rsm_exp, rsm_rb, rsm_vsv, rst_fail
critic_fail
critic_fail
NO_FAIL
CRITIC
rst_fail
st_app, st_assess, st_v_search, st_rb, st_vsv, st_tele, rsm_app, rsm_assess, rsm_v_search, rsm_rb, rsm_vsv, rst_fail
critic_fail
critic_fail
NO_FAIL
CRITIC
rst_fail

(a) UGV (b) UAV

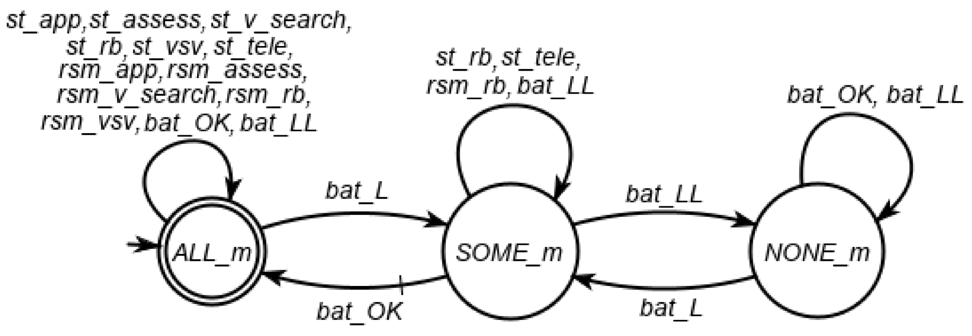
A.11 BATTERY LEVELS OPERATION MODES

The behaviors of the robots, showed on Figure 53, related to the three possible battery levels are similar to the simple and critical failures conditions. When the robots have battery OK, both types accept the execution of all maneuvers. By the moment the battery level gets LOW, both are able of executing teleoperation and return to base, and the UGV also can execute the VSV, being the prioritization choice on the TM implementation. And if the battery gets VERY LOW, the UGV disables all maneuvers, while the UAV still enables the safe land, since it is not part of the specification alphabet.

Figure 53 – Battery Levels Operation Modes.


st_app, st_exp, st_rb, st_vsv, st_tele, rsm_app, rsm_exp, rsm_rb, rsm_vsv, bat_OK, bat_LL
st_rb, st_vsv, st_tele, rsm_rb, rsm_vsv, bat_L
bat_OK, bat_LL
bat_L
bat_LL
ALL_m
SOME_m
NONE_m
bat_OK
bat_L

(a) UGV

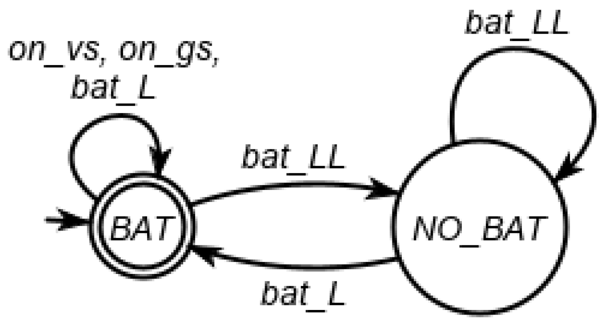

st_app, st_assess, st_v_search, st_rb, st_vsv, st_tele, rsm_app, rsm_assess, rsm_v_search, rsm_rb, rsm_vsv, bat_OK, bat_LL
st_rb, st_tele, rsm_rb, bat_LL
bat_OK, bat_LL
bat_L
bat_LL
ALL_m
SOME_m
NONE_m
bat_OK
bat_L

(b) UAV

A.12 SENSORS AT CRITICAL BATTERY

As the autonomous maneuvers are incapable of being implemented in case of a critical battery level, the possibility of activating the sensors is unnecessary, which would only consume unnecessary battery. Therefore, by the specification on Figure 54 we disable the activation of the UGV sensors if the battery is at a critical level. Since the UAV only has victim sensor as payload, and it is requires for the execution of safe land, such model is not applied for this robot type.

Figure 54 – Sensors at Critical Battery.

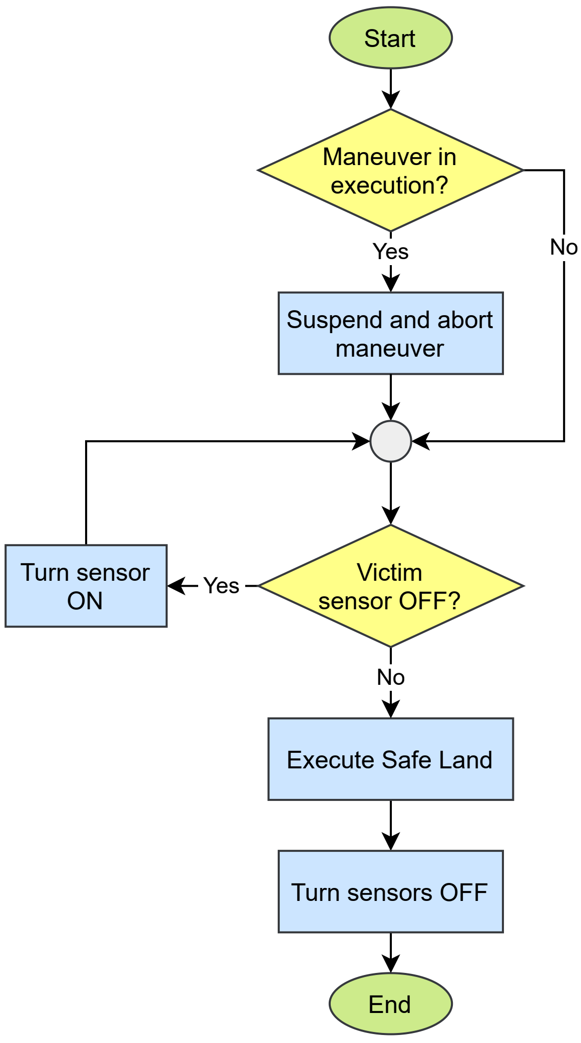


APPENDIX B – MODES OF OPERATION BEHAVIORS

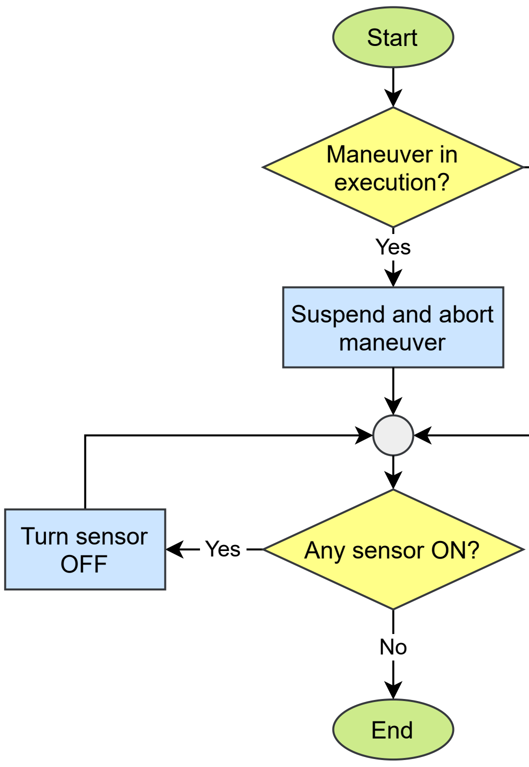
B.1 CRITIC SYSTEM

The CRITIC SYSTEM operation mode assigns to the robots the baseline sequences shown in Figure 55 aiming to safely stop the robots' actions without causing harm due to their failures. For the UGV, we can do this by simply aborting the actions the robot is performing in a way to make it stay still after the failure occurs. But for the UAV, since it works far from the ground, some extra precautions are required. Instead of just turning off the robot, we must ensure that it does not fall over a victim. So, after aborting the current task, the UAV starts the landing with the victim sensor ON, and if a victim is detected at a distance of less than 3 meters, the UAV moves in the opposite direction. The landing is finished only at a safe distance, and then the robot turns off all sensors and informs its position for further recovery.

Figure 55 – Baseline sequence expected for robots in Critical System operation mode.



(a) UAV.

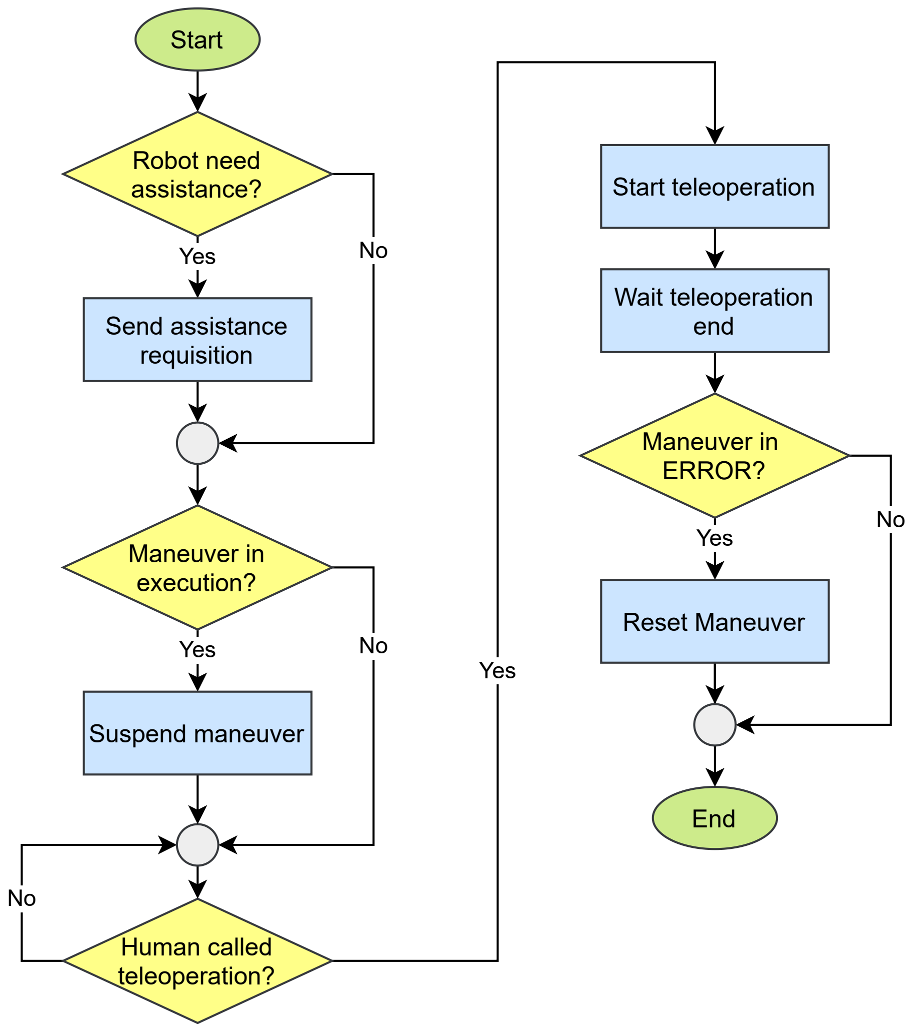


(b) UGV.

B.2 HUMAN-ROBOT INTERFACE

The Human-Robot Interface behavior is triggered in two specific situations. When the robot faces a situation that it verifies the need for help, and moments when humans require the teleoperation of the robot, e. g., when the Tech Specialist recognises something that he would like to have a better look at. When this mode of operation is selected, both types of robots attain the sequence presented in Figure 56. If the robot needs help, it will send an assistance requisition and wait for the human to require the start of the teleoperation. Otherwise, if the requisition comes from a human, the robot will suspend possible maneuvers in execution and start the teleoperation. When the teleoperation is accomplished, the robot will automatically reset the maneuver in failure if that is the case, and the conditions for maintaining the robot in this mode will not be satisfied anymore, changing it back to the previous behavior.

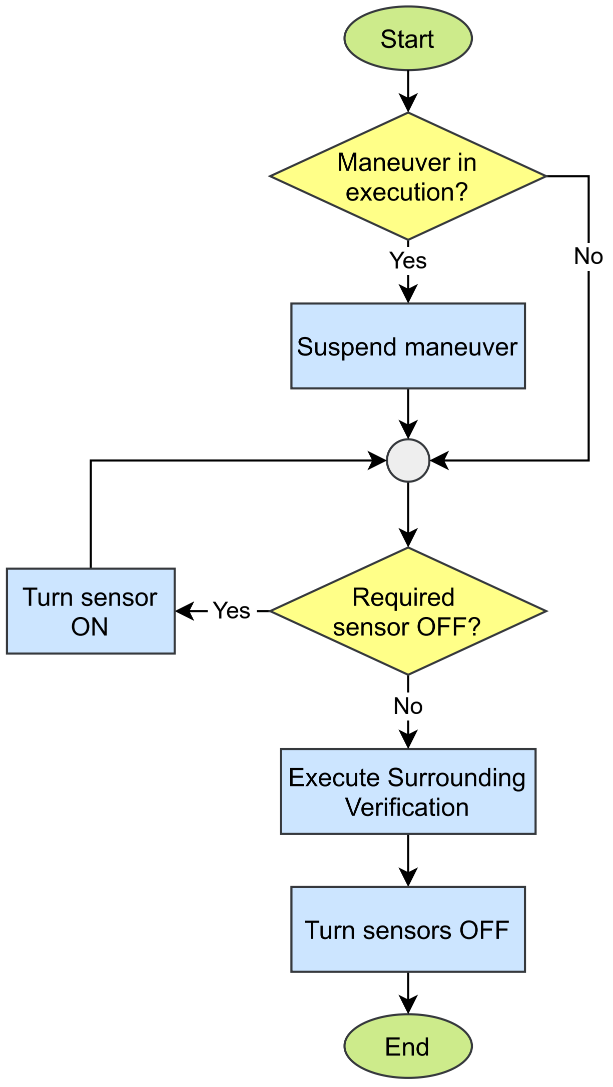
Figure 56 – Human-Robot Interface behavior characterised by the direct control of the robot's motion by the Tech Specialist.



B.3 VICTIM

If a victim is found by a robot, it will automatically change to this mode of operation with the objective of verifying the conditions of the environment around the victim. As can be seen in Figure 57, in this behavior, the robot won't abort the task being executed; it is only suspended while the verification of the victim's surroundings is accomplished. Before executing the VSV maneuver, the robot verifies if required sensors are turned on, such as the victim sensor for both robots and the gas sensor for the UGV. With this behavior, robots increase the Task Force Leader's situational awareness and increase the chances of finding victims since people tend to stay together in dangerous situations (ARNOLD; YAMAGUCHI; TANAKA, 2018).

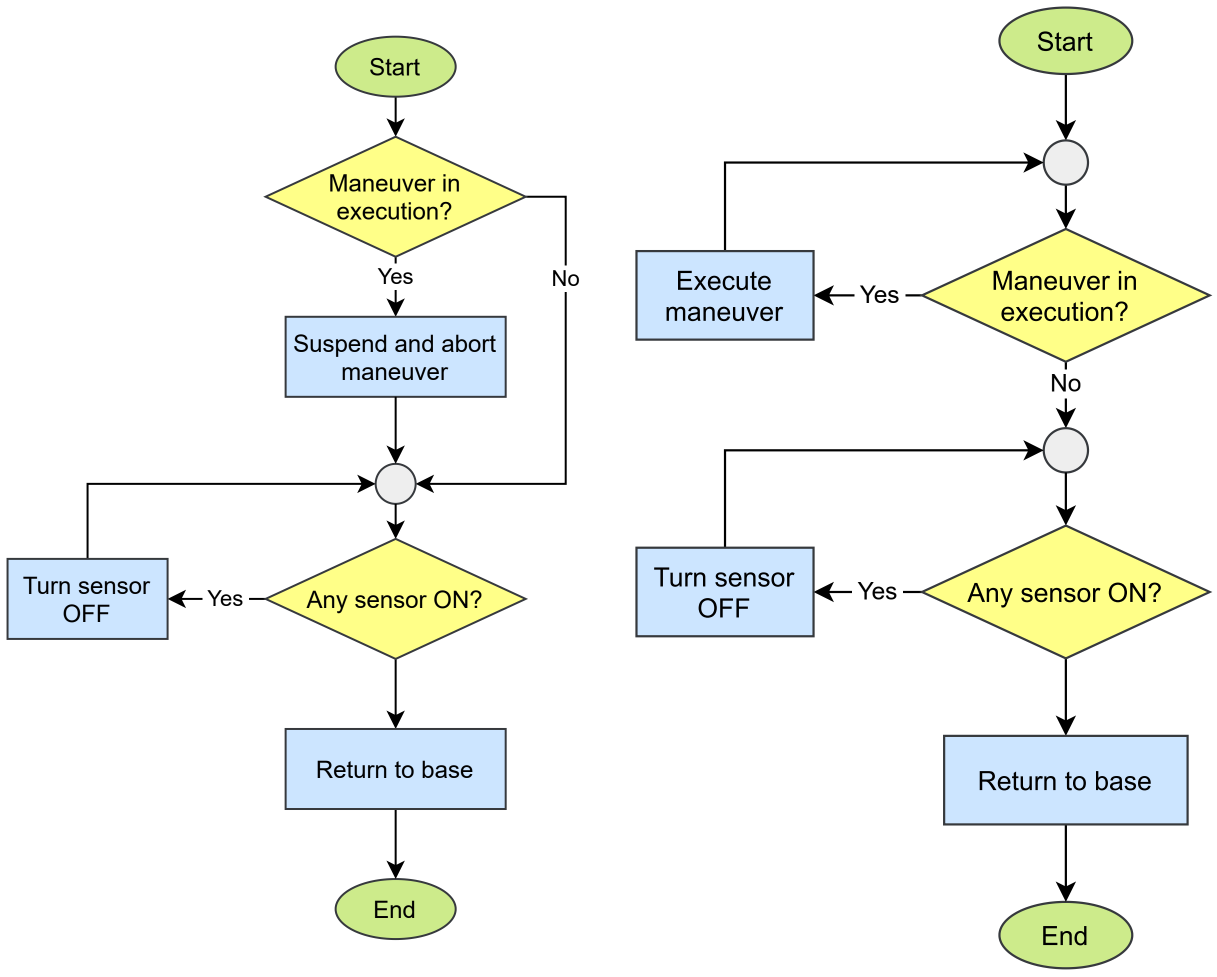
Figure 57 – Behavior responsible for making the robot verify the conditions around a victim. Which might increase the knowledge required for the Task Force Leader to define the best approach for the victim's extrication.



B.4 DEGRADED MODE I

This behavior represents the first level of degradation of the system, in which the robot becomes unable to accomplish some tasks but is still capable of moving autonomously. It is triggered by the occurrence of a simple failure or when the battery level gets low. Since both situations increase the robot's chances of getting stuck at the worksite, the expected behavior is that the robot returns to base for maintenance or battery recharge. Similarly to the Critic System mode of operation, here we designed slightly different behaviors for UAV and UGV, as shown in Figure 58. To reduce the risks, the UAV has to return to base immediately, and the UGV is still allowed to finish the task being executed before returning.

Figure 58 – Behavior assigned to the robots in the first possible level of degradation. UAVs abort the task being executed and immediately return to base, and UGVs are still capable of finishing the task.



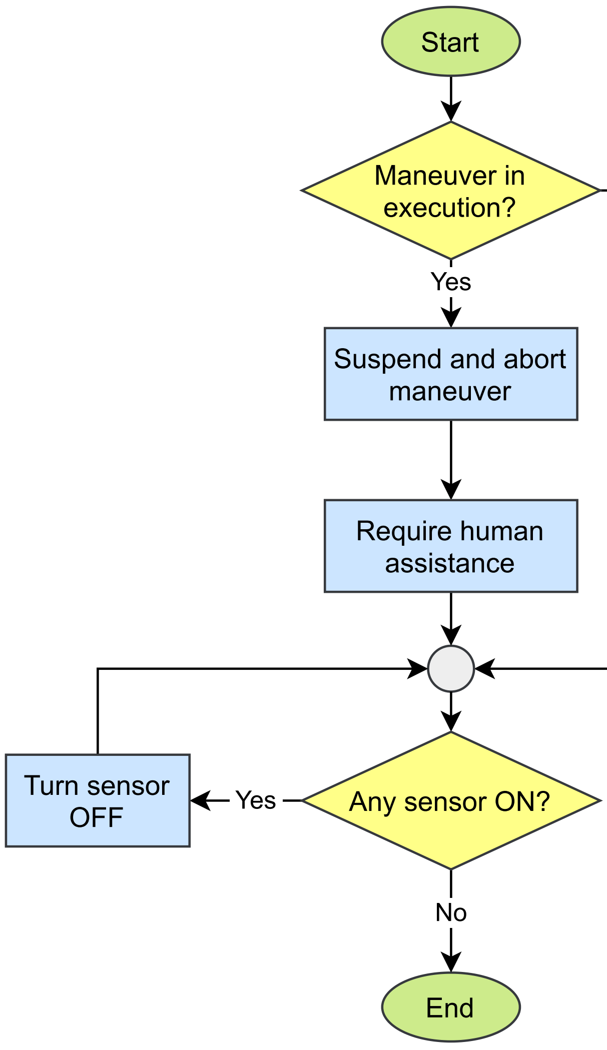
(a) UAV.

(b) UGV.

B.5 DEGRADED MODE II

This mode of operation was designed with the purpose of aborting tasks in the face of events that make them impossible to be carried out, e.g., motor or sensory impairments. The baseline sequence in Figure 59 shows that, unlike the other behaviors, the robot does not attempt to execute a maneuver. In this behavior, the robot only aborts the tasks being executed due to safety requirements, e. g., position failure makes autonomous movements dangerous, or if it becomes impossible to be accomplished, e. g., failure in victim sensor while executing search.

Figure 59 – More severe degraded mode of operation in which autonomous movements could be dangerous.



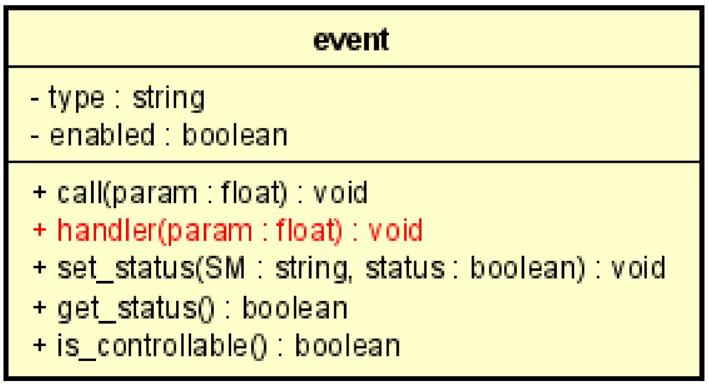
APPENDIX C – FRAMEWORK FOR TRANSLATING SUPERVISORS TO PYTHON

Here we summarize how formal models and supervisors synthesised by means of the SCT were implemented for the validation of this work in a simulated environment. For this objective, we developed a framework for the automatic translation of high-level models, represented by automata, into a Python implementation. It was necessary to guarantee the standardization of the implementation, reduce repetitive efforts, and avoid possible translation errors. The developed framework generates a logical sequence from XML files generated by the Supremica modeling tool, described by Malik et al. (2017).

To ensure the correct implementation of the supervisors, the system was implemented following the Supervisory Control System architecture, proposed by Queiroz and Cury (2002b). As was shown in Chapter 6, the SCS is composed of three main levels. The **Modular Supervisors (MS)** level is responsible for monitoring the plant dynamics and determining the set of disabled events. The **Product System (PS)** implements the high-level abstractions that mirror the real behavior of the system plant. And the **Operational Procedures (OP)** level is responsible for the interface between the high-level and low-level. Therefore, each of these levels was implemented in Python to ensure the correct system behavior.

The developed translation framework is fully available at SIMON (2021a) and is composed of the translation mechanism and the main levels of the SCS. Each of the modeled events is converted to a class abstraction specified in Figure 60. It is composed of some features required for the execution of the system and of a method called **handler**, in which the user can fill in the procedures to be executed in the real system as a result of triggering the event.

Figure 60 – Event class automatic generated by the translator for each event.


event
- type : string
- enabled : boolean
+ call(param : float) : void
+ handler(param : float) : void
+ set_status(SM : string, status : boolean) : void
+ get_status() : boolean
+ is_controllable() : boolean

In addition to generating the classes responsible for controlling each event, the automatic generator creates a translation table containing all possible events. By filling in this table, the user is informing the Supervisory Control System of the relationship between each high-level event and the corresponding low-level signals, which are interpreted by the Real System.

To make use of the proposed system, the user must follow the following procedures:

1. Generate the XML documentation of models developed in Supremica software;
2. Run the automatic code generator;
3. Fill in the table of equivalence between high and low level;
4. Fill in the desired procedures for each controllable event, in the handler method;
5. Execute the Supervisory Control System, consisting of the elements described below.

C.1 OPERATIONAL PROCEDURES LEVEL

The Operational Procedures level was implemented having as main components the 2 and 3 algorithms. The first, **Events Receiver** is a class that has a **receive** method responsible for making the conversions from low-level signals to high-level events that can be interpreted by the Product System[1]. Once this conversion is done, the *response* is inserted into an input buffer for non-controllable events, which is processed by the PS.

Algorithm 2 Events Receiver

```
1: converts the signal to **response**
2: insert the **response** on the input buffer of uncontrollable events
```

The **Event Handler** (Algorithm 3) has the opposite function to the Event Receiver. Its function is to convert *commands* triggered by the PS into operational sequences that can be interpreted by the Real System. Unlike the Event Receiver block, which is unique in the system, the Event Handler is abstracted by implementing the **handler** method of each controllable event, where the corresponding low-level procedures for each event must be found.

Algorithm 3 event_handler()

```
1: event parameters converts **commands** to procedures according translation_table
   and execute the procedures
```

[1] *Responses* linked to the occurrence of multiple signals or continuous conditions require the implementation of such equivalence in the Algorithm 2

C.2 PRODUCT SYSTEM LEVEL

The Product System is the main component of the proposed architecture, it has the function of executing a sequence of events arising from Real System signals (not controllable) or actions requested by the system (controllable). As quoted by Queiroz and Cury (2002b), it is essential that the states of the Plant and Supervisors are updated when each event occurs, in addition, it is important to give preference to the execution of non-controllable events, since they are signs that correspond to changes occurred in the Real System, impossible to be avoided.

This level was implemented through the Algorithm 4, which is executed as a*Thread* with a cycle linked to the presence of events in input *buffers* referring to controllable and non-controllable events. In order to execute this logic, it is necessary that the Product System has knowledge of the sub-plants and supervisors, which are previously converted from the XML representation to variables interpreted in Python.

The cycle implemented by the Product System starts by waiting for a new event, whether controllable or not. When identifying a new event, the algorithm first checks for the existence of non-controllable events and only in the absence of these it checks for possible controllable events, giving preference to the execution of the first one. If the selected event is non-controllable, the system just removes it from the *buffer*, not requiring the execution of any command. However, if the event is controllable, the algorithm must generate a command for the execution of operational procedures. This is done in line 7 by calling the **handler** method execution call of the corresponding event.

Algorithm 4 Product System

```
1:  while True do
2:    wait for new event on buffer
3:    if non-controllable event in buffer then
4:      event ← non-cont_events_buffer.pop(0)
5:    else if controllable event in buffer then
6:      event ← cont_events_buffer.pop(0)
7:      event_handler()
8:    end if
9:    for $i = 0;\ i < np;\ i++$ do
10:     plant[i].state_update(event)
11:   end for
12:   for $j = 0;\ j < ns;\ j++$ do
13:     supervisor[j].state_update(event)
14:   end for
15:   update historic of events
16: end while
```

After the event has occurred, in lines 9 to 14 the current status of the *np* sub-system plants and the *ns* supervisors are updated by executing the method shown in

Algorithm 5, thus generating a new set of enabled states and events. Finally, a variable that maintains the history of changes is updated, linking at each moment: the last event that occurred, events enabled after their occurrence and current status of the subsystems. It was specially important to this work so that the Task Manager could identify the current operating conditions.

The subsystems and supervisors are represented by the **StateMachine** and **Supervisor** classes, which have variables containing information on the states, events and transitions of the assigned models and have the **state_update** method, which follows the sequence presented in Algorithm 5. It initially checks if the event that occurred belongs to the model and if so, it updates the current status and enabled events.

Algorithm 5 state_update(event)

```
 1: wait for new event on buffer
 2: if event $\in$ alphabet then
 3:    update state
 4:    for e $\in$ alphabet do
 5:       if e $\in$ uncontrollable) || (e $\in$ transitions from current state) then
 6:          enable e
 7:       else
 8:          disable e
 9:       end if
10:    end for
11: end if
```