Safety vs. Efficiency: AI-Based Risk Mitigation in Collaborative Robotics

Ahmad Terra
GFTL ER AI
Ericsson AB
Stockholm, Sweden
e-mail: ahmad.terra@ericsson.com

Hassam Riaz
GFTL ER AI
Ericsson AB
Stockholm, Sweden
e-mail: hassam.riaz@ericsson.com

Klaus Raizer
GFTL ER AI
Ericsson Telecomunicacoes S.A.
Indaiatuba, Brazil
e-mail: klaus.raizer@ericsson.com

Alberto Hata
GFTL ER AI
Ericsson Telecomunicacoes S.A.
Indaiatuba, Brazil
e-mail: alberto.hata@ericsson.com

Rafia Inam
GFTL ER AI
Ericsson AB
Stockholm, Sweden
e-mail: rafia.inam@ericsson.com

***Abstract*—The use of AI-based risk mitigation is increasing to provide safety in the areas of smart manufacturing, automated logistics etc, where the human-robot collaboration operations are in use. This paper presents our work on implementation of fuzzy logic system (FLS) and reinforcement learning (RL) to build risk mitigation modules for human-robot collaboration scenarios. Risk mitigation using FLS strategy is developed by manually defining the linguistic values, tuning the membership functions and generating the rules based on ISO/TS15066:2016. RL-based risk mitigation modules are developed using three different Qnetworks to estimate the Q-value function. Our purpose is twofold: to perform a comparative analysis of FLS and RL in terms of safety perspectives and further to evaluate the efficiency to accomplish the task. Our results present that all the proposed risk mitigation strategies improve the safety aspect by up to 26% as compared to a default setup where the robot is just relying on a navigation module without risk mitigation. The efficiency of using FLS model is maintained to the default setup, while the efficiency of using RL model is reduced by 26% from the default setup. We also compare the computation performance of risk mitigation between centralized and edge execution where the edge execution is 27.5 times faster than the centralized one.**

Keywords-human-robot collaboration; risk mitigation; safety analysis; ISO/TS15066:2016; fuzzy logic system; reinforcement learning; automated warehouse; smart manufacturing

I. Introduction

Repetitive tasks can be easily replaced by robots to achieve automation specially in smart manufacturing, automated logistic management, healthcare, etc. Even though the field of robotics is getting more sophisticated every day, a simple task for a human may still be a challenge for a robot. As we can see in DARPA Robotics Challenge (DRC) 2015, robots were having difficulties to walk up the stairs, grasp a drill while standing, and maintain its balance while passing a door frame [1]. A solution to this could be a collaboration between humans and robots where humans perform more intelligent tasks and robots perform more repetitive tasks. However, when a robot is employed in the human's working place without any boundaries, a proper safety analysis needs to be performed.

ISO/TS15066:2016 defines safety requirements to enable robots working together with humans in a shared workspace [2]. A safety analysis can be implemented to gather the information about the environment, measure the risk of potential hazards and then execute necessary actions to ensure the safety of the operation. Risk management process [3] is used for safety analysis purposes and the use of artificial intelligence (AI) algorithms, specially Fuzzy Logic System (FLS) [4] is common in different risk management phases, e.g. risk analysis and risk mitigation [5], [6], [7], [8]. These works mainly focus on the use of a single algorithm with the perspective of providing safety.

This paper presents a comparative analysis of using different AI algorithms for risk mitigation and evaluate them with respect to safety and efficiency of the tasks allocated to robots. Our main focus is on using AI algorithms in risk mitigation for human-robot collaboration (HRC) scenarios in an automated logistic warehouse to have safe operations without compromising efficiency of the robot.

The main contribution is three-fold: First, we implement two different AI algorithms for the risk mitigation, which are

The authors acknowledge SCOTT (www.scott-project.eu) which has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway.

(1) fuzzy logic system (FLS) and (2) reinforcement learning (RL). Second, we present a comparative analysis for these algorithms with respect to safety and efficiency. Regarding to safety, we analyse the behaviour of the robot when it encounters an obstacle. The robot is expected to avoid the obstacle or reduce its speed when it is in a risky situation. In terms of efficiency, we measure the speed, duration, and path length of the robot while performing its tasks. Third aspect of our work is the computational performance comparison of a centralized system (where all the modules run on the robot's processor) to the edge execution. This is done to mitigate the increased computation time of heavy-load modules (that may require high computing power and consume most of the processor's resources) and thus resulting in poor robot's response time.

In our previous paper [6], we have proposed to use risk management process [3] for the safety analysis in HRC scenarios and presented the risk assessment in details without any implementation of AI. We used Hazard Operability (HAZOP) technique [9] to manually identify different roles of the humans in collaborative scenarios; defined the use cases of unsafe scenario, safety requirements, and safety recommendations. We further identified the risks and the possible hazards that may appear during the scenarios. We proposed to generate three concentric circles safety zones around the robot: critical (red) circle for critical zone, warning (yellow) and safe (green) zones that indicate the risk level of the obstacle with regards to its position/distance from the robot. The detailed implementation of risk analysis is presented in [7]. The first step in this process was identifying the environment to generate a scene graph information. The scene graph contains important properties such as type, distance, direction, and speed of the robot's obstacles. These properties are then used in risk assessment module to calculate the risk value based on the hazard and risk identification phase done in [6]. The FLS was used to calculate the risk value which implemented linguistic rules such as "*if the obstacle is static and the distance is near and the direction is in the front of the robot, then the risk value is very high*".

In this paper, we extend our work to implement different AI algorithms for risk mitigation and to perform comparison analysis of these algorithms with respect to safety and efficiency. The outline of the paper is organized as follows: Section II presents the related works of risk mitigation and collision avoidance for the mobile robots. Section III explains implementation details of FLS and RL algorithms. Section IV evaluates the obtained results and finally, Section V concludes the work and suggests future directions.

II. Related Work

Risk mitigation problem for a mobile robot is highly relevant with collision avoidance topics. A lot of work is performed using different AI algorithms (e.g. rule-based fuzzy logic, reinforcement learning, etc.) for this purpose.

FLS [4] provides high interpretability and easy understandability due to its rule-based nature as compared to other methods. Farkhatdinov et al. [8] utilized FLS to generate force feedback when a human is controlling a mobile robot. They controlled the force of the hand-controller based on the robot's distance to the obstacle. This implementation is partly similar to our implementation in mitigating the risk of the robot, however it is focusing only on the hand-controller aspect, not on the robot's movement

FLS for mobile robot navigation and collision avoidance were presented in [10], [11], [12] which directly control the left and right wheel speed of the robot. All of them took the distance to obstacle from the sensors for the input of their system. The work of [10] and [11] also took the information of the goal position for the input but the performance of their work was evaluated only using static obstacles. The work of Saleh et al. [12] implemented FLS for controlling four robot behaviours which are emergency stop, obstacle avoidance, wall following, and move to point. This work assigned specific sensors to specific behaviour and did not consider the type of the obstacle. The main differences of these works from ours are the input and the output of the system. We use scene graph information and consider the risk value of the obstacle. Our risk mitigation modules does not fully control the movement of the robot but scaling the wheels speed which are controlled by the default robot navigation.

FLS requires all rules and membership functions to be defined manually before performing its tasks. This may consume a lot of resources especially in fine-tuning the membership functions. RL algorithm such as Q-Learning [13], is another AI-based method in which defining these tasks are not required. A RL algorithm optimizes its model by executing actions and obtaining rewards and states during its operation.

A work of Faust et al. [14] presented a RL algorithm that has a comparable result as the state-of-the-art collision avoidance algorithm called Optimal Reciprocal Collision Avoidance (ORCA) [15]. They projected a high dimensional state space to a low dimensional feature space and only consider the closest obstacle. Their RL state was the position of the robot on the map and their model directly controlled the robot navigation. In our RL model, we use the default navigation system of the robot and formulate the state as the obstacle's properties around the robot.

The work of Coors [16] used an array of distance information gathered from laser scan sensor along with the distance and angle of navigation target to characterize the RL state. His experiment also employed Deep Q-Learning method but did not consider the type and the risk value of the obstacle.

Some other works, such as [17], [18], [19] developed Deep Q-Learning [20] models to avoid collision during robot operation. A work by Sullivan and Lawson [17] used raw image data from the robot's camera to predict three different actions (turn left, turn right, and go forward). Xie et al. [18] implemented two different networks to avoid an obstacle. The first network is used to predict depth information which is fed to the second deep q-network to determine the action of the robot. A work of Namba and Yamada [19] also used the raw image information, but aggregated with other information for the input of their deep q-network. All these works have just used camera data. On the other hand, instead of using raw images or array of distance information, we use the scene graph [21] and risk information as the input of risk mitigation module. Scene graph was chosen as it has semantic information and it concisely represents the important properties of the obstacle. An example of the scene graph information that is used in our work is shown in figure 3b. Please note that the method to produce a scene graph information is outside the scope of this paper and we use it as the input to the risk mitigation module. Another difference of our work from [17], [18] is the categorization of objects types (like human, static objects or dynamic objects) and then calculating risk based on the types which is not considered in these works. The risk value for humans is different than static or other dynamic objects.

III. AI-Based Risk Mitigation

This paper's main contribution lies in risk mitigation module as presented in Figure 1 where the default navigation modules are shown in the green boxes and the additional risk management modules are shown in the blue boxes. The risk mitigation module takes the risk value and the scene graph information as the inputs and generates speed scaling of the robot wheels as the output. It is developed with the intention to be independent and agnostic to the robot's navigation module. In order to do that, our risk mitigation calculates the adjustments that must be made in the speed values generated by the navigation module. This adjustment is essentially a factor that is multiplied by the current robot's wheel speeds. Here we name this factor as "speed scale" and is represented by a real value that ranges from 0 to 1.2 (a speed scale of 1 means that the robot executes its default operation). Figure 1 illustrates how the developed risk management modules works alongside a given navigation module. The following sections describe two risk mitigation strategies developed in this work: fuzzy logic and reinforcement learning.

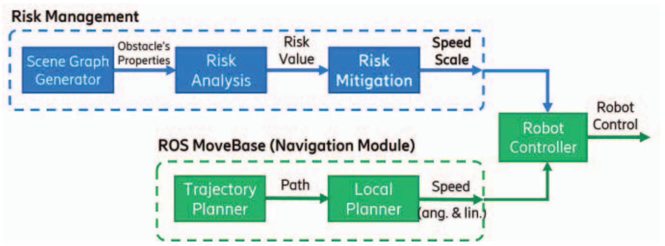


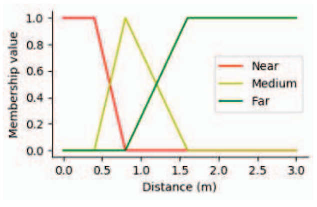
Figure 1. Robot control system where the default navigation modules are shown in the green boxes and the additional risk management modules are shown in the blue boxes.

A. Fuzzy Logic System (FLS)

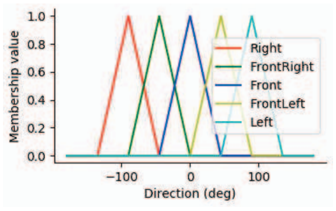
The implementation of FLS for risk mitigation only considers the scene graph of the riskiest obstacle in the robot's field of view. For instance, in Figure 3a, the FLS will only consider the human as the obstacle because he has the highest risk value due to 1) his distance to the robot (which is closer than the conveyor belt) and 2) his unpredictable behavior. The obstacle's distance, direction, and risk value are the properties of the scene graph used as the input variables to the FLS.

We categorize the distance variables into three linguistic values which are '*near*', '*medium*', and '*far*' as shown in Figure 2a. The trapezoidal functions are used as the membership functions (MFs) of '*near*' and '*far*' because they represent the comparison function such as less than (<) or greater than (>). Thus, a distance which is less than 0.4 meters would have a high membership value of 'near' category and a distance which is more than 1.6 meters would have a high membership value of '*far*' category. A triangular function is used for the 'medium' MF because it only covers a specific range of the distance (from 0.4 m to 1.6 m). This membership function can be modified depending on the robot's maximum speed —a slow robot may define a shorter distance for each linguistic value and vice versa—.

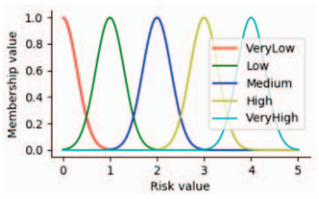
Five linguistic values ('*right*', '*front right*', '*front*', '*front left*', and '*left*') were used to classify the obstacle's directions. Every linguistic value used triangular MF and covered the equal space of direction as shown in figure Figure 2b. The robot uses this information to determine its direction when avoiding the obstacle. The risk value variable has five linguistic values which are *'very low', 'low', 'medium', 'high',* and *'very high'* and the membership functions are Gaussian as shown in Figure 2c.



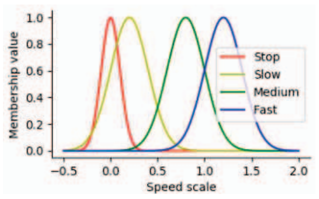
(a) MF for each membership value (*near, medium,* and *far*) of the obstacle's distance (m) variable.



(b) MF for each membership value (*right, front right, front, front left,* and *left*) of the obstacle's direction variable.



(c) MF for each membership value (*very low, low, medium, high,* and *very high)* of the obstacle's risk variable.

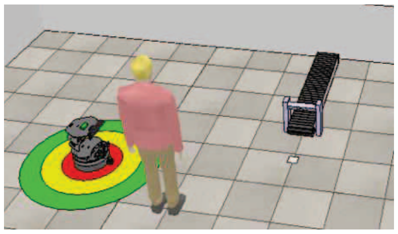


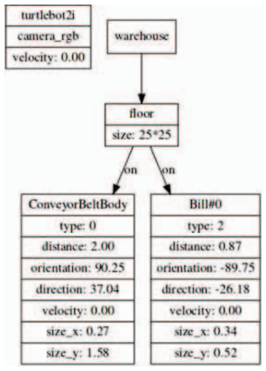
(d) MF for each membership value (*stop, slow, medium,* and *fast*) of the speed scale variable.

Figure 2. Fuzzy logic system membership functions for all linguistic variables and linguistic values used in this experiment.

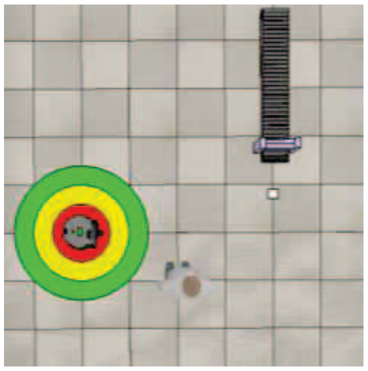
The scaling speed for the output of the FLS consists of four linguistic values which are *'stop', 'slow', 'medium',* and *'fast'*. Every linguistic value used Gaussian MF where the mean value is 0, 0.2, 0.8, and 1.2 for *'stop', 'slow', 'medium',* and *'fast'* respectively. The standard deviation value of '*stop*' membership function is 0.1 while the others are 0.2 as shown in Figure 2d.

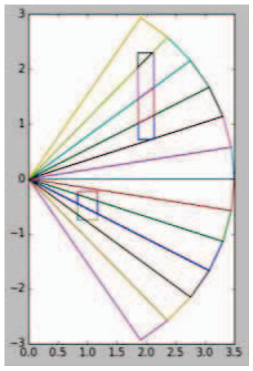
In order to generate the output, the FLS applies all the input values to the specified rules. The rules are easy to understand because this system uses linguistic variables and values. An example of the rules for the risk mitigation system is "*if obstacle's distance is near and its direction is*


(a) A situation where the robot with its safety zone encounter a human and static obstacle during the operation.



(b) Scene graph generated from the scenario shown in Fig. 3a


(c) A scenario similar to Fig. 3a as seen from the top.



(d) Identified obstacles are mapped into the robot's FoV.

Figure 3. The scene graph information summarizes the properties of the obstacles in front of the robot. Figure 3d shows the result of scene graph parsing to construct 2D map from the example shown in Figure 3b.

on the front then the left wheel scale is stop and the right wheel scale is stop". This example shows that the module mitigates the risk by stopping the robot's movement to avoid a collision. The mitigation rules[1] using FLS were created manually based on safety recommendations of ISO/TS 15066 [2] and are presented in [6].

B. Reinforcement Learning (RL)

Another algorithm that we use to build risk mitigation module is reinforcement learning. As opposed to the FLS, this algorithm does not require rules generation process. This algorithm relies on the state formulation and it is optimized by the reward function which measures the quality of the action. The following subsections explain the detail of our RL implementation for the risk mitigation modules

1) State Formulation: The proposed RL-based risk mitigation module is designed to consider all obstacles that are visible in the field of view (FoV) of robot's camera. We explain it in Figure 3. The scene graph is parsed to create a 2D map that incorporates the size and the position of each obstacle (see Figure 3b). We consider only those obstacles which are within 3.5 m range as farther distance is considered safe. The FoV of 114° is equally split into 12 zones as shown in Figure 3d. We first measure the distance from the robot to the nearest obstacle in every zone and then use this set of distances as the input state of the RL model.

Other information are also added as the input features such as *nearest obstacles distance, nearest obstacles direction, radius of warning zone, radius of safe zone, maximum risk value, robots linear speed,* and *robots angular speed*. The first two properties are only applied in the Hybrid Q-Network while the last five properties are used in all RL models. These two values are used in the the Hybrid Q-Network because the zone distance information are convolved to their neighbouring zone distance while the other models can infer these information directly from the input.

2) Reward Function: The quality of the action chosen by the RL model is measured by a reward. For risk mitigation, the reward function is defined in Algorithm 1. This function calculates the reward value (R) depending on the executed action and the obstacle's positions. The importance of the obstacle is measured by the clearance reward ($R_{clearance}$) which is reciprocal to the distance between the robot to the nearest obstacle ($d_{nearest}$); closer obstacles induces a higher clearance reward. The yaw reward (R_{yaw}) gives a positive reward if the model chooses an action that avoids the obstacle. Otherwise, a negative reward is given with respect to the percentage of angular speed ($v_{angular}$) of the robot. The multiplication of clearance reward ($R_{clearance}$) and yaw reward(R_{yaw}) indicates the quality of the model in avoiding the obstacle. This multiplication is only considered when there is an obstacle inside the safety zones (*critical, warning,* or *safe zone*). To induce the robot to keep moving when the situation is safe, a positive reward is given if the robot has moved 0.017 m. The notation of $r_{critical}$, $r_{warning}$, and r_{safe} in Algorithm 1 represent the radius of *critical, warning,* and *safe* zones consecutively.

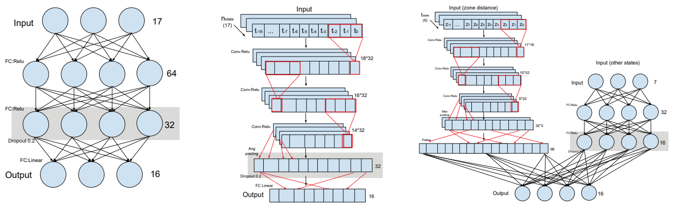
3) Actions: The actions for the RL model consist of four discrete scaling speed values which are 0.0, 0.4, 0.8 and 1.2 for each wheel (in our case, left and right wheels). They represent the *'stop', 'slow', 'medium',* and *'fast'* linguistic values in the FLS. Since the scaling speed is applied individually to the wheel, we have sixteen different combinations of the actions.

[1] The complete rules for risk mitigation using FLS can be found in: https://github.com/EricssonResearch/scott-eu/blob/master/simulationros/src /turtlebot2i/turtlebot2i safety/src/mitigation rules.py

Algorithm 1 Reward function calculation

```
$R_{clearance} \leftarrow \frac{1}{d_{nearest}}$
if avoiding obstacle then
    $R_{yaw} \leftarrow +1$
else
    $R_{yaw} \leftarrow -\frac{v_{angular}}{Max(v_{angular})}$
end if
if collision happens then
    $R_{yaw} \leftarrow -5000$
else if $d_{nearest} < r_{critical}$ then
    $R \leftarrow (R_{clearance} \times R_{yaw}) - 50$
else if $d_{nearest} < r_{warning}$ then
    $R \leftarrow (R_{clearance} \times R_{yaw}) - 10$
else if $displacement > 0.017$ then
    if $d_{nearest} < r_{safe}$ then
        $R \leftarrow (R_{clearance} \times R_{yaw}) + 1$
    else
        $R \leftarrow +1$
    end if
else
    $R \leftarrow -1$
end if
```

4) *Q-Network Architectures:* We develop three different Q-Network models for the risk mitigation module to estimate the quality of the state and the action. The first model implements a fully connected network (FCN) as shown in Figure 4a. It has 17 features for the input state and two hidden layers with rectified linear unit (ReLU) activation function. The first and second hidden layers have 64 and 32 neurons, respectively. There is also a dropout (20%) layer before the output layer. Since this model is built to measure the quality of the actions, the output layer is a fully connected network with 16 neurons and linear activation function.



(a) FCN model. (a) CNN model. (a) Hybrid model.

Figure 4. The illustration of the deep Q-network architectures.

The second model implements convolutional layers and records the last twenty states as the input. Each state has the same features as the FCN model (with 17 information for each state) which is then transposed so the convolution process occurs in a time-step manner. This model is illustrated in Figure 4b with the details of the network shown in Table I. We call this model as the CNN model for simplicity in the rest of the paper.

TABLE I. THE CONFIGURATION OF 1D CONVOLUTIONAL NEURAL NETWORK FOR THE Q-NETWORK

No.	Layer	Activation	Size	Filter size	Filter stride
1	Convolutional	ReLU	32 Filter	3	1
2	Convolutional	ReLU	32 Filter	3	1
3	Convolutional	ReLU	32 Filter	3	1
4	Average pooling	-	-	-	-
5	Fully connected	Linear	16 Neurons	-	-

The hybrid model is the combination of FCN and CNN architectures. This model takes the latest six data of the twelve zone distance states for the input of the convolutional layer. The input of the fully-connected network layer are the other seven features mentioned in the second paragraph of Section III-B1. In this model, every convolutional step processes the information between zone-distance with their neighbouring zones. The last layer of this network is connected to both network (the CNN part and the FCN part) as shown in Figure 4c.

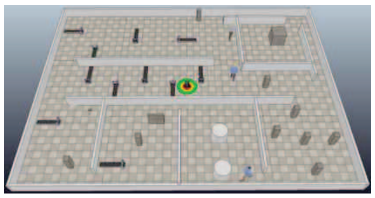
TABLE II. THE CONFIGURATION OF THE HYBRID MODEL FOR THE Q-NETWORK

No.	Layer	Activation	Size	Filter size	Filter stride
1	Convolutional	ReLU	16 Filter	2	1
2	Convolutional	ReLU	32 Filter	2	1
3	Convolutional	ReLU	32 Filter	2	1
4	Max pooling	-	3	-	-
5	Fully connected	ReLU	32 Neurons	-	-
6	Fully connected	ReLU	16 Neurons	-	-
7	Fully connected	Linear	16 Neurons	-	-

C. *Automated Warehouse Scenario*

The AI-based risk mitigation methods are trained (in case of RL strategy) and tested in automated warehouse scenarios modeled in V-REP simulator [22]. We have used the same computer as the edge device (specified in table III) to run the simulator program. The usage of a simulator simplifies the environment setup and avoids physical damage during development phase. The simulated warehouse is formed by different objects: conveyor belts, boxes of different sizes and shapes, walls and humans that are walking around freely. The simulated robot is based on Turtlebot2i model equipped with 3D camera and LIDAR. Figure 5a shows the simulated warehouse.

The trajectory planner module for the navigation has an occupancy grid map [23] to generate the path. This map only contains the walls of the warehouse so other obstacles such as humans or boxes must be detected by the scene graph generator during the operation. The robot navigates towards random positions during the training to explore various conditions that may appear. Figure 5a shows the simulated warehouse and its occupancy grid map is shown in Figure 5b.


(a) Simulated warehouse with humans and unknown static obstacles, such as boxes and conveyor belts.

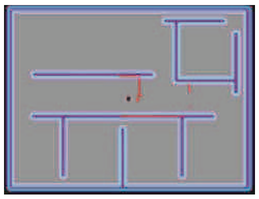
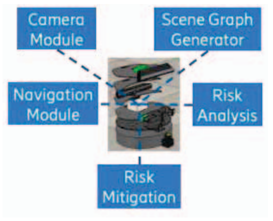

(b) The occupancy grid map only consists of wall of the warehouse.

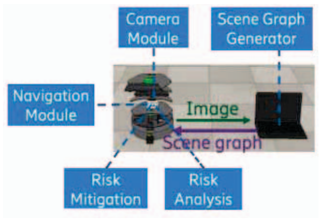
Figure 5. The simulated warehouse and its occupancy grid map for training the reinforcement learning model.

D. Computation Architecture

This part presents our experiment to evaluate the computation performance of two different architectures to execute the risk management system on the real robot. We do not evaluate the behaviour of the risk mitigation on the real robot and leave it as a future development. One centralized method for realrobot implementation is to run all the modules on the robot's processor. There are five main modules to run the experiment which are navigation, camera, scene graph generator, risk analysis and risk mitigation modules. This method is also known as local computation and it is illustrated in Figure 6a.



(a) A centralized local computation architecture where all modules run on the robot's processor.



(b) An edge computation architecture where the scene graph generator is transferred to an edge device.

Figure 6. The illustration of local and edge computation architectures.

Other configuration that we investigated is the edge computation. In this setting, we transfer the scene graph generator — which requires Mask R-CNN [24] model to extract the features of the obstacle before constructing the scene graph— to an edge device that has more processing power. Our work utilize ROS [25] modularity to distribute the computing load between two different processing units. The edge device used in this experiment is suitable for running a deep neural network such as Mask R-CNN [24] because it has a discrete graphical processing unit. The communication between the robot and the edge device is done through a wireless network. The robot sends both of RGB and depth images that are captured by the sensor to the edge device. The edge device processes these information and send a scene graph information to the robot. The specification of the robot's processor and edge device for this experiment are shown in Table III.

TABLE III. SPECIFICATION OF THE COMPUTING DEVICES

Machine	Robot's Processor	Edge Device
Processor	Intel Celeron J3455 2MB cache, 2.3 GHz	Intel i7-7700HQ 6MB cache, 3.8 GHz
Memory	4GB, DDR3L-1600MHz	16GB, DDR4-2400MHz
GPU	Intel HD Graphics 500	NVIDIA GTX 1060 Max-Q (6GB GDDR5)
Storage	240 GB SSD	1TB HDD
Connectivity	802.11AC WiFi/	802.11AC WiFi/
OS	Ubuntu 16.04	Ubuntu 16.04

The router's specification that we used in our experiment is given below:

• Model	: Sagemcom WiFi Hub C1.
• Max download speed	: 1000 Mbps.
• Max upload speed	: 100 Mbps.
• LAN port	: 4 RJ45 100/1000 Mbps.
• WiFi	: 2.4 GHz b/g/n.

IV. EVALUATION

Here we evaluate the efficiency and safety aspects of the implemented AI algorithms and compare the performance of two different computation architectures.

A. Evaluation Setup

This following subsection describes how the evaluation is carried out.

1) Risk Mitigation: The risk mitigation algorithms are evaluated in the same concept as the work of Ceballos and Velasquez [26] for mobile robot navigation system. However, we define different metrics and categorize them for safety and efficiency aspects. The safety metrics can be classified into three different concepts which are: the time percentage of the obstacle in the safety zone; the speed and risk values; and the clearance distance to the obstacles. The details of the metrics to evaluate the safety and efficiency aspects are explained in Table IV and Table V consecutively.

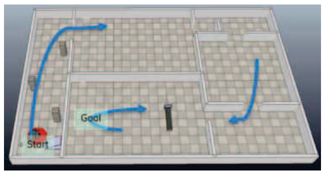
In order to test the generalization of the AI-based risk mitigation models, the evaluation scenarios have different layout from the training scenario. To evaluate the behaviour of the algorithms when they interfere with different types of obstacles, we created four scenarios which are:

- **Scenario 1**: Warehouse without unknown obstacles as shown in Figure 7a.
- **Scenario 2**: Warehouse with eight unknown static obstacles as shown in Figure 7b.
- **Scenario 3**: Warehouse with seven humans as shown in Figure 7c.
- **Scenario 4**: Warehouse with seven humans and eight unknown static obstacles as shown in Figure 7d.

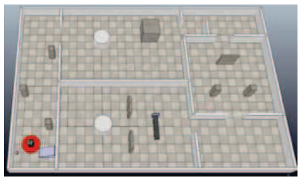
In Scenario 1, the planning module relies on a map containing all obstacles so the robot will not collide with them. This scenario is used to evaluate how the developed risk mitigation deals with the known obstacles. However, in the actual operation, a robot may drop its load and human operators may obstruct the robot's path. These kind of obstacle must also be avoided by the robot. Thus, Scenarios 2 and 3 are designed to evaluate the ability of risk mitigation to avoid unknown static obstacles and humans, respectively. The Scenario 4 is the combination of Scenarios 2 and 3. In every scenario, the robot will follow this path: *start - goal - start goal* to evaluate the AI algorithms as seen in Figure 7a.

TABLE IV. THE METRICS TO EVALUATE THE SAFETY ASPECTS

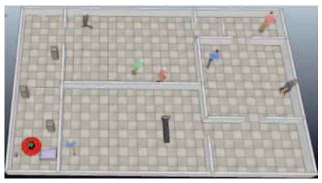
Metric	Description
% of safe zone	Time percentage that the closest obstacle remains in the warning zone
% of warning zone	Time percentage that the closest obstacle remains in the warning zone
% of critical zone	Time percentage that the closest obstacle remains in the critical zone
Mean risk value	Mean of the risk value during operation
Maximum risk value	Maximum value of risk that is experienced during the operation
Mean of (risk×speed)	This metric collects/sums the multiplication of the robot's speed and its risk value and then divides it by the count of recorded data
Max of (risk×speed)	This metric records the maximum multiplication of the robot's speed and its risk value during the operation
Mean distance to obstacle (m)	Mean of every distance sensor (e.g. LIDAR) from robot's center to its surrounding during the operation
Mean minimum distance to obstacle (m)	Every distance sensor (e.g. LIDAR) record their minimum measurement during operation. This metric takes the mean of that data from all sensors
Minimum distance to obstacle	Minimum distance to the closest obstacle that the robot has ever experienced during the operation
Number of collisions	Counts the collisions that happened during the operation



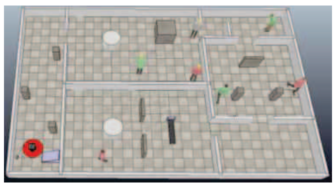
(a) Scenario 1: Warehouse with known obstacles.



(b) Scenario 2: Warehouse with the unknown static obstacles.



(c) Scenario 3: Warehouse with humans.



(d) Scenario 4: Warehouse with humans and unknown static obstacles.

Figure 7. Different warehouse scenarios for evaluating AI algorithms for risk mitigation.

TABLE V. THE METRICS TO EVALUATE THE EFFICIENCY ASPECTS

Metric	Description
Time execution	Duration needed by the robot to accomplish the task measured in seconds.
Path length	Traveled distance in meters to accomplish the task.
Mean distance to goal	Mean of the distances from each robot position to goal position, taken along the planned path. Lower values of this metric implies that the robot can reach the goal faster.
Mean speed	Mean of the robot's speed during operation.

2) Computation Performance: We measure the computation performance by recording the duration that each module takes to generate its output. The total duration of the risk management module comprehends from the time scene graph generator receives the information from the camera module until it outputs the speed scale as illustrated in Figure 8.

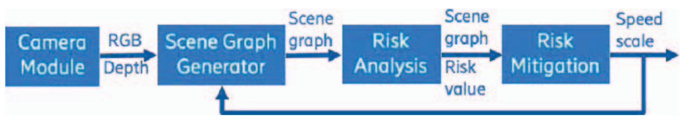


Figure 8. The data pipeline to record the total duration of risk mitigation.

B. Results

Here we present and discuss the results obtained from the experiments[2].

1) Comparison of AI-based risk mitigation methods with respect to safety and efficiency: Table VI shows the evaluation results of Scenario 1 where the best result for every metric is presented in **bold**. First column of the table presents the metric, second column presents results for the default operation without risk mitigation (this is calculated to make a comparison with different algorithms). Third, forth, fifth and sixth columns present results of using FLS, FCN, CNN and Hybrid models respectively. In this scenario, all operations that implement risk mitigation modules has higher value of the time-percentage for the robot in keeping the obstacle outside the critical and warning zones as compared to the default operation without risk mitigation. The time-percentage of critical zone metric is less in each module. The FCN model has the highest mean risk value (i.e. 0.542) because the robot operates within a close distance to the obstacles (FCN has the lowest value of mean distance to obstacles). However, the FCN model has a lower value of *mean of (risk×speed)* than the normal operation without risk mitigation. It indicates that the robot lowered down its speed when it encountered a high risk obstacle.

[2] The videos of the simulation can be found in: https://www.dropbox.com /sh/fkc6ugflaj27dl3/AACqxu8xXJvXDfvZJt1NM7oma?dl=0

TABLE VI. THE EVALUATION RESULT OF SCENARIO WITH KNOWN STATIC OBSTACLES IN SAFETY ASPECT

Parameter	w/o RM	FLS	FCN	CNN	Hybrid
Safe zone (%)	92.214	92.895	93.498	**94.927**	92.373
Warning zone (%)	7.645	7.069	6.431	**5.048**	6.506
Critical zone (%)	0.141	0.036	0.071	**0.026**	1.121
Mean risk value	0.469	**0.402**	0.542	0.434	0.449
Max risk value	2.544	2.476	2.803	**2.347**	3.144
Mean of (risk×speed)	0.210	0.202	0.203	**0.146**	0.211
Max of (risk×speed)	1.086	1.340	1.093	**0.898**	1.531
Mean dist. to obst (m)	2.525	2.539	2.517	2.5311	**2.654**
Mean min dist to obs (m)	0.531	**0.547**	0.516	0.328	0.349
Min dist to obst (m)	0.233	0.234	0.226	**0.255**	0.200
Collision	0	0	0	0	0

The Scenario 2 has both of known and unknown static obstacles. The FLS and the CNN model have significantly improved the time-percentage of safe zone metric by 7% as compared to the default operation without risk mitigation. However, the safest operation in this scenario is achieved by FLS model which minimizes the time-percentage of critical zone metric to 0.65%. The CNN model significantly reduced the mean of (risk×speed) value by 26% and reduced the mean of risk value from 0.498 (without risk mitigation) to 0.369. The operation of the CNN model tends to reduce the robot's speed and it has the lowest value of mean speed metric (see Table X).

There was a collision during the operation when the hybrid model was used for the risk mitigation module. This model has more 'aggressive' behaviour than other models as shown by the highest mean speed metric and the lowest improvement of staying in the safe zone. It also has the highest mean and maximum of (risk×speed) value. This result shows that the robot did not reduce the speed during risky situation. The complete evaluation results for this scenario are shown in the Table VII.

TABLE VII. THE EVALUATION RESULT OF SCENARIO WITH UNKNOWN STATIC OBSTACLES IN SAFETY ASPECT

Parameter	w/o RM	FLS	FCN	CNN	Hybrid
Safe zone (%)	69.615	77.242	73.639	**77.869**	71.405
Warning zone (%)	22.858	22.103	22.518	**20.801**	24.350
Critical zone (%)	7.527	**0.655**	3.843	1.330	4.246
Mean risk value	1.132	**1.022**	1.236	1.116	1.113
Max risk value	3.071	**2.773**	2.932	2.988	3.368
Mean of (risk×speed)	0.498	0.484	0.479	**0.369**	0.548
Max of (risk×speed)	1.603	1.567	1.316	**1.165**	1.985
Mean dist. to obst (m)	2.174	2.141	**2.180**	2.161	2.166
Mean min dist to obs (m)	0.307	**0.366**	0.348	0.314	0.256
Min dist to obst (m)	0.200	0.231	0.224	**0.254**	0.200
Collision	2	**0**	**0**	**0**	1

Table VIII shows the evaluation result for the scenario with humans and known obstacles. In this scenario, all the developed risk mitigation modules improved the time-percentage of staying in safe zone. In terms of time-percentage of critical zone metric, the hybrid model has higher value than the normal operation while the other models have lower value. The FLS model has the highest improvement in the time-percentage of safe zone metric and the FCN model significantly reduced the time-percentage of critical zone metric. Similar to the previous scenario, the CNN model has the lowest mean and max of (risk×speed) values among the implemented AI algorithms.

TABLE VIII. THE EVALUATION RESULT OF SCENARIO WITH HUMANS IN SAFETY ASPECT

Parameter	w/o RM	FLS	FCN	CNN	Hybrid
Safe zone (%)	83.212	**87.297**	85.351	86.088	85.722
Warning zone (%)	14.999	11.349	14.549	13.108	**11.280**
Critical zone (%)	1.787	1.354	**0.101**	0.804	2.998
Mean risk value	0.980	**0.816**	0.950	0.901	0.893
Max risk value	**3.267**	3.682	3.606	3.680	3.670
Mean of (risk×speed)	0.421	0.330	0.317	**0.282**	0.352
Max of (risk×speed)	1.687	1.479	1.306	**1.272**	1.945
Mean dist. to obst (m)	2.501	2.449	2.484	2.498	**2.583**
Mean min dist to obs (m)	**0.437**	0.316	0.329	0.230	0.218
Min dist to obst (m)	**0.225**	0.200	0.200	0.200	0.200
Collision	**0**	1	**0**	2	**0**

Scenario 4 tried to get closer to the real automatic warehouse operation where both humans and unknown static obstacles were involved. In this scenario, all of the implemented AI algorithms reduced the time percentage of the robot having an obstacle inside the critical zone and improved the timepercentage of safe zone metric. The CNN model improved the time-percentage of safe zone metric by more than 9% and reduced the time-percentage of critical zone metric by 7%. In contrast, the hybrid model has the least improvement in the time-percentage of critical zone and the time-percentage of safe zone metrics.

Even though the FCN model has the highest mean risk value, the behaviour of this algorithm was safer than that in the hybrid model, FLS, and normal operation without risk mitigation. With a significant gap of (risk×speed) metrics, FCN and CNN models have notably safer operation than the other models. The detail evaluation results for this scenario are shown in the table IX.

In terms of efficiency, the FLS model slightly improved the performance of the robot operation. In three evaluation scenarios, the FLS model has the highest mean speed and the lowest duration among other algorithms. On the other hand, the CNN model performed the operation with the least efficiency. Even though it has the safest operation, the CNN model has the lowest mean speed and the longest duration. In the extreme case, the CNN model of risk mitigation module generated 26% slower operation and extended the time execution by 78%.

TABLE IX. THE EVALUATION RESULT OF SCENARIO WITH HUMANS AND UNKNOWN STATIC OBSTACLES IN SAFETY ASPECT

Parameter	w/o RM	FLS	FCN	CNN	Hybrid
Safe zone (%)	63.672	66.288	67.801	**72.835**	65.753
Warning zone (%)	27.224	28.802	28.145	**24.784**	25.252
Critical zone (%)	9.104	4.911	4.055	**2.381**	8.995
Mean risk value	1.327	1.380	1.535	1.395	**1.260**
Max risk value	3.670	3.666	3.669	**3.653**	3.658
Mean of (risk×speed)	0.569	0.566	0.462	**0.438**	0.526
Max of (risk×speed)	3.549	1.800	**1.219**	1.356	2.006
Mean distance to obstacle (m)	2.139	2.100	2.145	**2.187**	2.132
Mean min dist to obstacle (m)	0.203	0.286	0.212	**0.288**	0.214
Min dist to obst (m)	0.200	0.200	0.200	0.200	0.200
Collision	5	4	**0**	**0**	1

Generally, the FLS for risk mitigation provides an explainable decision where the variables, values and rules are represented in linguistic terms. The behaviour of the robot is also guaranteed to follow the specified rules which are generated from an extensive work and a detailed analysis. The number of rules will increase exponentially if we want to consider more obstacles as the input of the system. Besides that, every linguistic variable must be accompanied with a set of linguistic value and membership function.

TABLE X. THE EVALUATION RESULT OF THE EFFICIENCY ASPECT

	w/o RM	FLS	FCN	CNN	Hybrid
Scenario with known static obstacles					
Duration (sec)	123.05	**120.78**	127.83	168.06	142.98
Path length (m)	96.67	97.16	**95.44**	96.19	101.62
Mean distance to goal (m)	6.565	6.657	6.553	6.946	**6.358**
Mean speed (m/s)	0.444	**0.456**	0.424	0.325	0.434
Scenario with unknown static obstacles					
Duration (sec)	143.34	**136.57**	149.86	192.80	141.30
Path length (m)	103.07	102.74	**99.50**	101.99	103.48
Mean distance to goal (m)	6.633	6.481	6.506	6.713	**6.391**
Mean speed (m/s)	0.423	**0.455**	0.410	0.324	0.479
Scenario with humans					
Duration (sec)	**289.75**	377.81	445.04	516.50	512.78
Path length (m)	97.22	99.05	96.47	**96.76**	104.09
Mean dist to goal (m)	**6.639**	7.025	6.770	7.045	7.448
Mean speed (m/s)	**0.454**	0.420	0.385	0.329	0.352
Scenario with humans and unknown static obstacles					
Duration (sec)	355.36	**354.89**	596.61	614.06	504.66
Path length (m)	104.98	103.85	105.39	**103.59**	109.66
Mean distance to goal (m)	6.883	6.814	6.686	6.761	**6.412**
Mean speed (m/s)	0.418	**0.424**	0.331	0.309	0.393

The RL method allows risk mitigation module to consider more obstacles without generating more rules to cover all the possible situations. This method heavily depends on the state formulation and reward function to perform its intended task. Both components are the keys to optimize a RL model where there is a trade off between efficiency and safety aspect. Our experiment in implementing reinforcement learning for risk mitigation achieved the safest operation because the implemented reward function prioritized the safety aspect over the efficiency. In comparison to the FLS, it has lower efficiency because there are only sixteen discrete actions for the system (each wheel has four different speed scale values) rather than a continuous output as produced by the FLS.

The architecture of the network is also important because the DQN [20] method replaces the Q-matrix with a neural network. The FCN model considers only the latest values of the input and applies 1.2 scaling speed for several situations when mitigating the risk. The CNN and hybrid models take several previous information to select the action and the changes of the mitigating actions are not as frequent as the FCN model. Their actions are dominated by slower (≤0.8) scaling speed value when mitigating the risk.

2) Comparison of centralized vs edge executions: The test to measure the duration of each module is run with 100 iterations and the average values are shown in Figure 9. We obtained that the process to generate a scene graph information required around fifteen seconds in a centralized computation. This performance is not suitable for risk mitigation because a collision may have already happened during the process of generating the scene graph information. However, it only took 0.3 seconds in average to generate the scene graph on the edge device. In this architecture, the duration of risk analysis module was reduced by 45%. The risk mitigation with RL decreased the processing time by 36% in average while the FLS module has 14% less running time than the centralized implementation. Figure 9 shows the complete comparison between local and edge computation architecture.

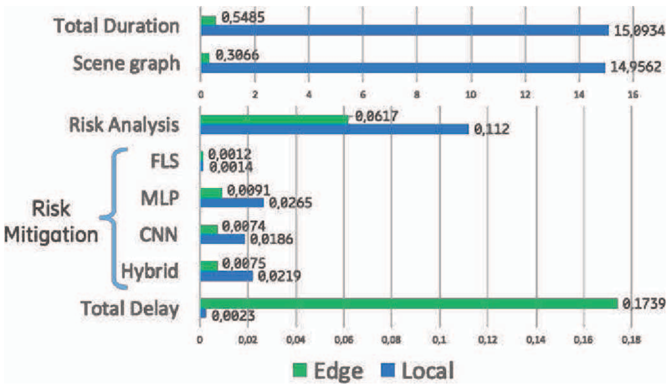


Figure 9. The duration of local and edge computational time in seconds.

V. CONCLUSIONS AND FUTURE WORKS

We have presented comparative analysis between fuzzy logic system and reinforcement learning algorithms for robot's efficiency and has the fastest computation time. On the other hand, the RL modules has safer operation than the FLS. However, the improvement of the safety comes with a cost of efficiency reduction —the safest operation has 26% slower mean speed than the operation without risk mitigation—. Choosing which risk mitigation algorithm to be used can be done by referring to the required safety and efficiency key performance indicators (KPIs). For example, risk mitigation module with FLS can be chosen when the warehouse must deliver a rapid operation because it maintains the efficiency. On the other hand, the RL model can be used when the warehouse is busy and have more humans in the shared space to increase the safety of the operation.

In terms of computation architecture, the centralized system is not suitable for mitigating the risk. The process to generate the scene graph information takes too much time. Collision with an obstacle may not be avoided if response time is 15 seconds or more. However, moving the scene graph generator module to an edge device not only provides a reasonable response time, but also decrease the running time of other modules.

The future continuation of this work may investigate the effect of capturing more information by increasing the camera's field of view. A centralized scene graph generator that processes images from the warehouse camera (e.g. CCTV) can also be investigated. In terms of RL, the actions of the model can be extended by adding more discrete actions or changing it into continuous values. Other possible development is to build the risk mitigation model after the last layer of the scene graph generator model (Mask R-CNN [24]). In this setting, one network will produce the scene graph information while also mitigating the risk.

REFERENCES

[1] C. Atkeson, B. P. Wisely Babu, N. Banerjee, D. Berenson, C. Bove, X. Cui, M. Dedonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, and X. Xinjilefu, *What Happened at the DARPA Robotics Challenge Finals*, 04 2018, pp. 667–684.

[2] ISO, *ISO/TS 15066:*2016*: Robots and robotics devices - Collaborative robots*. Geneva, Switzerland: International Organization for Standardization, Feb. 2016.

[3] ISO, *ISO 31000:2018: Risk management Guidelines*. Geneva, Switzerland: International Organization for Standardization, Feb. 2011.

[4] L. A. Zadeh, "Fuzzy sets, fuzzy logic, and fuzzy systems," G. J. Klir and B. Yuan, Eds. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1996, ch. Fuzzy Sets, pp. 19–34.

[5] R. Inam, E. Fersman, K. Raizer, R. Souza, A. N. Junior, and A. Hata, "Safety for Automated Warehouse exhibiting collaborative robots," in *28th European Safety and Reliability Conference (ESREL'18)*. Trondheim, Norway: IEEE, June 2018, pp. 2021–2028, available at https://www.taylorfrancis.com/books/9781351174657.

[6] R. Inam, K. Raizer, A. Hata, R. Souza, E. Forsman, E. Cao, and S. Wang, "Risk assessment for human-robot collaboration in an automated warehouse scenario," in *2018 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 09 2018, pp. 743–751.

[7] A. Hata, R. Inam, K. Raizer, S. Wang, and E. Cao, "AI-based safety analysis for collaborative mobile robots," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2019, pp. 1722–1729.

[8] I. Farkhatdinov, J. Ryu, and J. Poduraev, "Rendering of environmental force feedback in mobile robot teleoperation based on fuzzy logic," in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA)*, Dec 2009, pp. 503–508.

[9] J. Guiochet, "Hazard analysis of humanrobot interactions with HAZOPUML," *Safety Science*, vol. 84, pp. 225–237, 04 2016.

[10] H. Omrane, M. Masmoudi, and M. Masmoudi, "Fuzzy logic based control for autonomous mobile robot navigation," *Computational Intelligence and Neuroscience*, vol. 2016, pp. 1–10, 09 2016.

[11] N. H. Singh and K. Thongam, "Mobile robot navigation using fuzzy logic in static environments," *Procedia Computer Science*, vol. 125, pp. 11 – 17, 2018, the 6th International Conference on Smart Computing and Communications.

[12] S. Zein-Sabatto, A. Sekmen, and P. Koseeyaporn, "Fuzzy behaviors for control of mobile robots," *Journal of Systemics, Cybernetics and Informatics*, vol. 1, 02 2003.

[13] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, UK, May 1989. [Online]. Available: http://www.cs.rhul.ac.uk/~chrisw/new thesis.pdf

[14] A. Faust, H. Chiang, N. Rackley, and L. Tapia, "Avoiding moving obstacles with stochastic hybrid dynamics using PEARL: PrEference Appraisal Reinforcement Learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 484–490.

[15] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, "Reciprocal *n*-body collision avoidance," in *Robotics Research - The 14th International Symposium, ISRR 2009, August 31 - September 3, 2009, Lucerne, Switzerland*, 2009, pp. 3–19. [Online]. Available: https://doi.org/10.1007/978-3-642-19457-3 1

[16] B. Coors, "Navigation of mobile robots in human environments with deep reinforcement learning," Master's thesis, KTH, School of Computer Science and Communication (CSC), 2016.

[17] K. Sullivan and W. Lawson, "Deep obstacle avoidance," *IJCAI-16 Workshop on Deep Learning for Artificial Intelligence*, 2016.

[18] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," in *RSS 2017 workshop on New Frontiers for Deep Learning in Robotics*, 2017.

[19] T. Namba and Y. Yamada, "Risks of deep reinforcement learning applied to fall prevention assist by autonomous mobile robots in the hospital," vol. 2(2):13, 2018. [Online]. Available: https: //www.mdpi.com/2504-2289/2/2/13/pdf

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[21] W. Liao, M. Y. Yang, H. Ackermann, and B. Rosenhahn, "On support relations and semantic scene graphs," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 131, 09 2016.

[22] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1321–1326.

[23] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525

[24] K. He, G. Gkioxari, P. Dollr, and R. Girshick, "MaskR-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2980–2988.

[25] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, 01 2009.

[26] N. M. Ceballos and J. V. Velasquez, "Benchmark framework for mobile robots navigation algorithms," *REVISTA FACULTAD DE INGENIERA*, vol. 23, p. 65, 01 2014.