Unifying Control and Verification of Cyber-Physical Systems (UnCoVerCPS)

WP5 Realisation of Cyber-physical Systems

D5.3 - Report on Application Simulation Data and Experimental Results

WP5	**D5.3 - Report on Application Simulation Data and Experimental Results**
Authors	Mark Burgin (RUR) Joshué Pérez (Tecnalia) Maria Prandini (PoliMi) Alexander Rauch (Bosch) Jens Oehlerking (Bosch)
Short description	The following document discusses the test results obtained from simulation and real-work trials for each of the use cases developed withing the UnCoVerCPS project.
Deliverable type	Report
Dissemination level	Public
Delivery date	December 2018
Internal review by	Xavier Fornari (ET) Maria Prandini (PoliMi) Daniel Heß (DLR) Olaf Stursberg (UKS) Zonglin Liu (UKS)
Internally accepted by	Matthias Althoff (TUM)
Date of acceptance	Dec 28, 2018

Document history:

Version	Date	Author/ Reviewer	Description
0.1	Sept 25, 2018	M Burgin	Created a skeleton report
0.2	Nov 26, 2018	M Burgin	Version for internal review
0.3	Dec 14, 2018	M Burgin	Version for internal acceptance
1.0	Dec 28, 2018	M Burgin	First Formal Issue

Contents

1 Introduction **6**

2 Automated Driving Use Case **8**

2.1 Introduction ... 8
2.2 Modelling of the problem ... 9
2.2.1 Nominal trajectory planning approach ... 10
2.2.2 Cooperative overtaking scenario ... 11
2.3 Simulation results ... 12
2.3.1 Scenario and tools ... 12
2.3.2 Results ... 13
2.4 Real vehicle results ... 16
2.4.1 Scenario and tools ... 16
2.4.2 Results ... 16
2.5 Real vehicle in cooperative scenarios ... 19
2.5.1 Scenario and tools ... 19
2.5.2 Results ... 20
2.6 Conclusions ... 22

3 Smart Grid Use Case **23**

3.1 Description of the smart grid configuration ... 24
3.2 Energy management problem ... 28
3.3 Proposed data-driven solution ... 30
3.3.1 Control policy parametrization ... 30
3.3.2 Scenario solution ... 32
3.3.3 PV energy production profiles reconstruction ... 33
3.4 Performance assessment of the proposed energy management strategy ... 38
3.5 Concluding remarks ... 40

4 Human-robot collaboration use case **42**

4.1 The GRAIL Simulation ... 43
4.1.1 GRAIL robot ... 44
4.1.2 Kinect sensors ... 45
4.1.3 Light curtains ... 49
4.1.4 Conveyors ... 49

4.1.5 Ingredients and bases 50

4.1.6 Gripper 50

4.2 The GRAIL testbed 51

4.3 The application of the UnCoVerCPS online verification approach 52

4.3.1 The verifier 52

4.3.2 Testing the verifier in simulation 55

4.3.3 Real-world testing of the verifier 61

4.3.4 Conclusions from testing the verifier alone 66

4.4 Increasing the operational efficiency in the presence of human co-workers 67

4.4.1 The plan selector 67

4.4.2 Plan selector co-worker motion prediction 69

4.4.3 Plan selector evaluation function 69

4.4.4 Optimising the plan selector weightings in simulation 72

4.4.5 Testing the plan selector performance in simulation 78

4.4.6 Testing the plan selector performance in the real world 80

4.4.7 Conclusions from testing the plan selector 80

4.5 Meeting the required safety integrity level 81

5 Mobile Robotics Use Case 84

6 Conclusions 85

7 Appendix A - Human-Robot Distance Calculation 86

References 96

1 Introduction

This report describes the final testing performed, and the results obtained, from experiments on the use cases explored within the UnCoVerCPS project. Experimental results are presented from both simulation models and real-world trials of the use cases described below.

Section 2 of this report describes an automated driving use case which employs a trajectory planning and decision module based on the on-line verification framework developed within UnCoVerCPS. Different manoeuvres, such as lane keeping, lane changing, and merging among multiple automated vehicles, have been considered. These manoeuvres have been defined in the decision module, assuming cooperation between the vehicles. Experimental results from simulation, mixed real-world and simulation, and fully real-world trials are presented using a collaborative lane change manoeuvre as an example.

Section 3 describes a smart grid use case, in which models of the smart grid components are derived based on first principles and experiments. Based on this model, a disturbance compensator enforcing certain reachability specifications has been designed using a data-driven approach to characterize the disturbance. This use case relates to the work under WP 1 and WP 2, on modeling and control, respectively. In contrast to most other UnCoVerCPS results, however, a probabilistic approach is adopted for control design since disturbance is modeled as a stochastic process and reachable sets are characterized by a probability level. Results are presented from real-world trials using a testbed consisting of a photo voltaic panel installation, a storage battery, a load simulator and a grid interface.

Section 4 describes a human-robot collaboration use case, which focusses on a system to collaboratively assemble food products such as sandwiches or pizzas. This use case employs the UnCoVerCPS on-line verification techniques developed in WP 2. It is tested in both simulation and on a real-world implementation to assess both its safety and efficiency in the presence of human co-workers. Further work is also described which improves the efficiency of operation of the system in the presence of nearby co-workers.

Section 5 summarizes the work done on a mobile robots use case at Bosch, which is described more fully in Deliverable D5.2 [2]. Bosch, together with TUM, have evaluated the UnCoVerCPS online verification methods on the basis of a newly introduced use case of mobile service robots. Thereby, a conformant pedestrian model was identified to ensure transference of verification results into reality based on the techniques developed in WP 1 and WP2. Section 5 also summarizes recent work where the conformant pedestrian model has been extended towards a new “jogger” model in order to develop a proof-of-concept for conformance monitors within a simulation environment.

In the concluding section, the benefits obtained from the novel methods developed within the UnCoVerCPS project are summarized.

2 Automated Driving Use Case

2.1 Introduction

New advances in automated driving seek to reduce the risk in driving situations, such as overtaking. A study by the US National Highway Traffic Safety Administration [32] attributed the cause of 94% of vehicle crashes to human behaviour, and among these 33% to mistakes or wrong decisions during manoeuvring. The last couple of decades have seen substantial progress in the field of advanced driver assistance systems (ADAS). Currently, automated driving (AD) is developed to advance the ADAS and many people hope that human-related risk factors can be mitigated with the advent of this technology.

In recent decades significant advances have been achieved in the field of AD, such as the improvement of trajectory tracking controllers [24], path planning [19], wireless communication [7], and perception. Many challenges remain due to the difficulty of introducing the technology into a complex, unstructured and unpredictable environment, which is often governed by social norms and interactions between a multitude of participants [26]. The authors of [23] have conducted a study based on information given by the Bureau of Transportation Statistics in the United States (2015), where it is demonstrated that a fleet of 100 automated vehicles driving for 24 hours a day and all year round will require around 8 years of testing to demonstrate a rate of 77 injuries per 100 million persons; and 500 years to obtain a rate of 1.09 fatalities per 100 million persons (nominal data obtained from the injuries and fatalities caused during vehicle crashes reported in the United States in 2013).

Recent research propagates formal methods for automated driving as one possible solution to the problematic situation of validation by means of test driving. In [29] a formal approach was proposed to verify that distributed vehicle control systems based on Adaptive Cruise Control (ACC) satisfy the safety objective. In [6] a method for automated vehicles safety verification based on on-line reachability analysis has been presented.

For our AD use case, Tecnalia and DLR have developed a trajectory planning and decision module based on the on-line verification framework in UnCoVerCPS. Different manoeuvres, such as lane keeping, lane changing, and merging among multiple automated vehicles, have been considered. These manoeuvres have been defined in the decision module, assuming cooperation between the vehicles. Therefore, the trajectory planning algorithm must be fast enough to allow sufficient time for both planning and verification in one update cycle.

Verifying every manoeuver made by automated vehicles is a demanding and difficult task. An example is shown in Figure 1, where three vehicles are considered in a two-way road

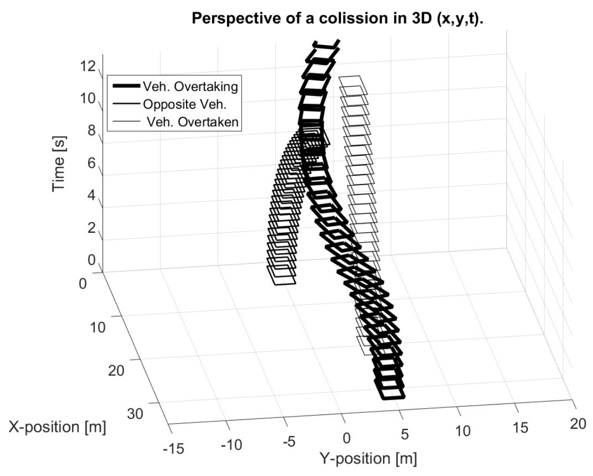


Figure 1: Collision in XY plane vs time.

overtaking manoeuvre, with a potential collision between the overtaking vehicle and opposite vehicle. This figure shows the trajectory of each vehicle for position and time. In this case (overtaking scenario), the existence of a possible collision must be considered at any time. We demonstrate a cooperative method to verify the manoeuver in real time. Our method is fast enough to perform each planned trajectory online.

The rest of this section is organized as follows: Section 2.2 describes the problem of cooperation among automated vehicles. Section 2.3 shows the simulation tool used and results obtained. These are later validated in Section 2.4 and 2.5, using real platforms and virtual vehicles. Finally, our conclusions are shown in Section 2.6.

2.2 Modelling of the problem

Our approach for the verification of the automated driving process is based on an online analysis of each vehicle's actions. An overview of the proposed system architecture is given in Figure 2. The nominal planning module provides the driving skills required to autonomously navigate in different traffic situations. The environment information is received frequently from a set of perception modules. The nominal planning module predicts the evolution of the traffic situation and determines actions to safely achieve the vehicle's goals. The actions considered are: driving along a lane, following a vehicle, merging into a gap in real traffic, and stopping at an intersection.

However, there might be some situations in which the planned trajectory is unsafe. Therefore, a verification module analyses the safety of each action before it is executed by a controller [20]. The verification module uses worst-case models for other traffic participants as well as the ego vehicle. Using these worst-case models, emergency manoeuvers are planned,

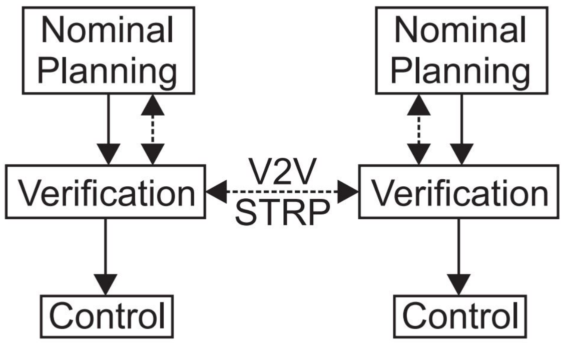


Figure 2: General architecture under cooperative maneuvering.

which bring the vehicle to a standstill. This is true for every combination of behaviours of the other traffic participants as well as all ego measurement errors and disturbances. If a safe emergency manoeuvre is found, branching off the intended plan at position P, the current action is forwarded to the control module for execution until arriving at P. Otherwise, the emergency manoeuvre is executed.

The nominal planning module and the verification module operate at fixed time steps. They receive new sensor information for the i-th update at time t_i, and first compute a nominal action and then the emergency manoeuvre in a time interval of $[t_i, t_i + T_p]$. The resulting nominal action takes place during the time interval $[t_i + T_p, t_i + 2T_p]$ and the i-th emergency manoeuvre could start at $t_i + 2T_p$. Based on the reaction time of human drivers, the delay of $2T_p$ influences the degree of conservatism of this approach. The longer the delay, the more distance must be maintained between consecutive vehicles. In such a case, the time T_p must be large enough to compute an acceptable trajectory for the nominal conditions and small enough to avoid any possible dangerous condition when executing the emergency manoeuvres.

2.2.1 Nominal trajectory planning approach

The overall lane following and lane changing optimization is a non-convex and non-linear problem. In our approach, the solution space to lane following and lane changing manoeuvres have been restricted to a convex sub-problem that can be efficiently solved.

Both longitudinal and lateral motion plans are obtained below, as a solution of an optimal control problem (OCP) for a third-order integrator chain. The same notation will be used for the lateral and longitudinal case. Our model can be described mathematically as:

$$d_{x,y} = \iiint J_{x,y}(t)\, dt\, dt\, dt, \tag{1}$$

where the indices $< x, y >$ are used to represent the longitudinal and lateral domain, respectively. The state-space model of (1) is:

$$\begin{bmatrix} \dot{d}_{<x,y>} \\ \dot{v}_{<x,y>} \\ \dot{a}_{<x,y>} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_{<x,y>} \\ v_{<x,y>} \\ a_{<x,y>} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} J_{<x,y>}, \tag{2}$$

where $d_{<x,y>}$ refers to position, $v_{<x,y>}$ is speed, $a_{<x,y>}$ is acceleration and $J_{<x,y>}$ is jerk. The state vector can be defined as $x = [d, v, a]^T$. The optimization process will be weighted by a vector $\omega \in \mathbb{R}^4$, that can be solved by a standard quadratic programming problem. Additionally, the problem will be discretized in equidistantly spaced points of time $t_0, t_1, ..., t_N$ with $t_{i+1} - t_i = T_s$.

2.2.2 Cooperative overtaking scenario

Vehicle to vehicle communication enables automated vehicles to exchange their intentions and to cooperate in different situations. Some approaches transmit the specific trajectories [27] of each cooperating vehicle, which can be data intensive. Instead, it may suffice to exchange constraints, which decouple the trajectories of all involved entities. Based on [20], we propose a simple spacetime reservation protocol, which allows vehicles to negotiate constraints to solve conflicting traffic situations.

A cooperative exchange involves one request and confirmation. Per exchange, one request message is sent, and one confirm message per answering vehicle is sent back. The cooperative exchange is initiated by an imminent conflict in vehicle positions, e.g., when a vehicle V_R intends to execute a manoeuvre such as changing to or crossing over a higher priority lane. V_R must consider possible conflicts originating from future actions of vehicle V_P on the targeted higher priority lane. If V_R's intended manoeuvre is not feasible under the given predictions, it broadcasts a request message to cancel the message. Most importantly, it specifies a lane area to be reserved for V_R, as indicated in Figure 3: a part of the lane starting at a coordinate x_0, y_0, with a length l_R along the lane is reserved for a time interval $[t_0, t_1]$.

The confirmation message, on the one hand, obliges V_P to avoid any overlap with the reserved area. This can be achieved by adding corresponding constraints to V_P's manoeuvre planner. On the other hand, V_P's confirmation allows V_R to relax the constraints imposed by the prediction of V_P's intent, according to the negotiated reservation.

The spacetime reservation protocol is integrated into the above nominal manoeuvre planning as follows: the committing vehicle adds a virtual vehicle of length L_R to the targeted lane object set during the time interval $[t_0, t_1]$, starting at any position given by $< x_0, y_0 >$

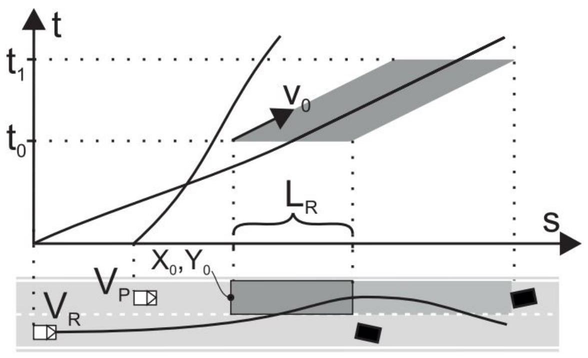


Figure 3: Reserved area in s and t for a cooperative lane change.

with a velocity v_0.

2.3 Simulation results

A simulation environment is used for testing the cooperative overtaking scenario (described in Deliverable D5.2, Appendix D). Then, real tests have been performed with the DLR and Tecnalia vehicles.

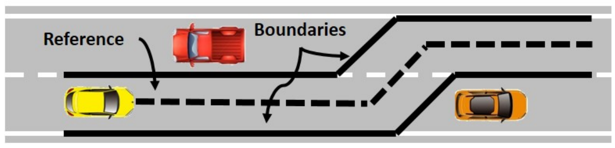
2.3.1 Scenario and tools

A first test case has been evaluated in a simulation environment using a straight two-way road, presented in [27]. We use a hybrid approach, based on parametric curve and MPC, first solving the regular lane following task, and then the lane change task based on a lateral offset double integrator chain with MPC, as in [27]. The longitudinal behaviour is defined with a triple integrator chain. An example scenario is shown in Figure 4.

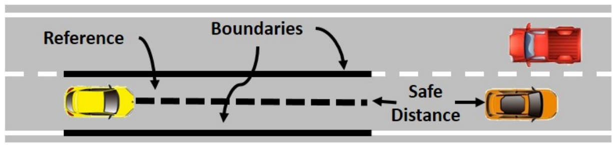
Our simulator is based on a kinematic model of the vehicle. The future prediction states are evaluated considering unsafe conditions. In the overtaking manoeuvre the boundaries and the trajectory reference are known, based on the information of the last iteration.

Figure 4(a) shows the overtaking use case when the left lane is free and the right lane is blocked. The boundaries are set in terms of lane occupation considering the vehicle's width. The reference is located in the middle of these boundaries. Figure 4(b) shows overtaking when the right lane is blocked by an oncoming vehicle. Both boundaries are in the right lane, and the maximum distance is limited to a safety distance between the ego vehicle and the vehicle in front. Finally, Figure 4(c) shows the boundaries for returning to the original lane, due to an oncoming vehicle in the left lane. The total distance is reduced to avoid a collision with the vehicle in front while keeping a safe distance.

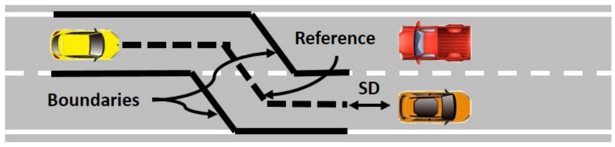
This approach permits verifying the trajectory planning results, avoiding any frontal



(a) Overtaking without any blocking.



(b) Overtaking with vehicles blocking.



(c) Started overtaking and frontal vehicle

Figure 4: Boundaries and reference in different situations.

Figure 5: Tecnalia's Twizy platform (left) and Dynacar model (right).

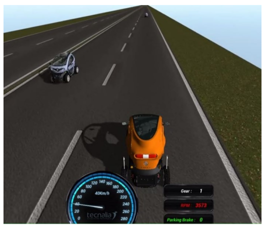
collision and improving the traffic flow during the lane change manoeuvre.

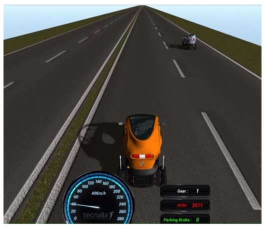
The simulator used for the test is Dynacar by Tecnalia, which permits the precise modelling of the vehicle dynamics. For the simulations, we have used a model of a Renault Twizy that is Tecnalia's platform for automated driving applications (Figure 5).

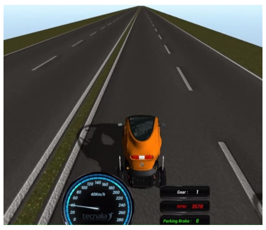
2.3.2 Results

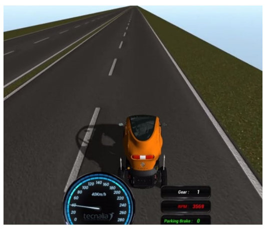
Our approach has been tested with three vehicles in two different scenarios: i) an overtaking manoeuvre with enough distance to finish it, and ii) an overtaking manoeuvre without enough distance to finish it.

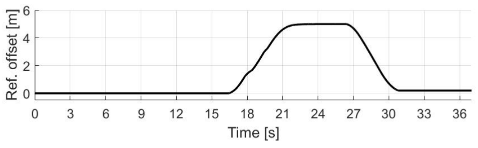
The first scenario includes 3 vehicles, where the first is the automated vehicle (the


(a) Passing.

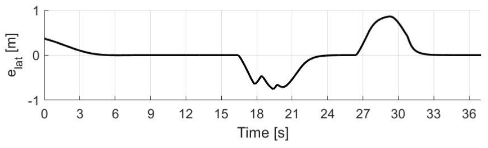

(b) Taking the other lane.


(c) Overtaking.

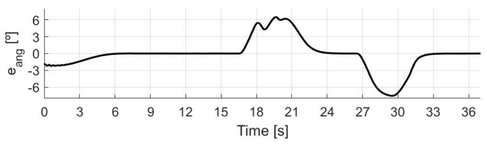

(d) Returning.



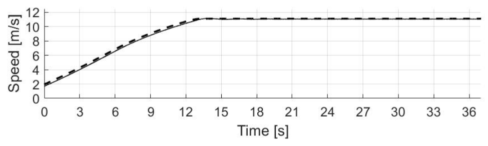
(e) Lateral offset.



(f) Lateral error.



(g) Angular error.



(h) Vehicle and reference speed.

Figure 6: Results of test case 1.

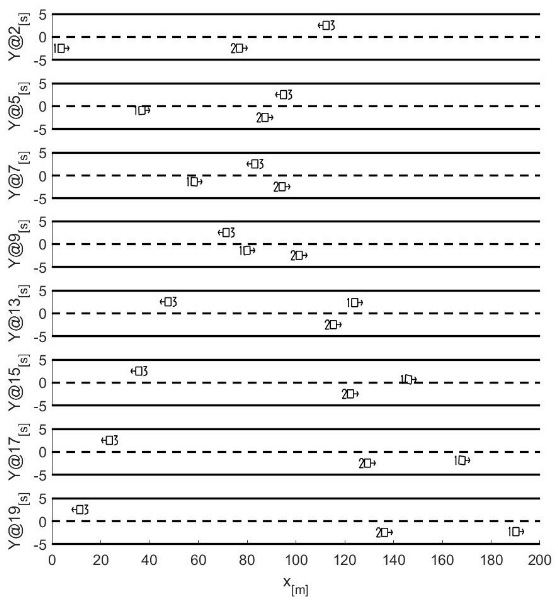
overtaking vehicle), the second vehicle (blocking) is driving at low speed, and the third vehicle is moving in the opposite direction (left lane).

Figures 6(a) to 6(d) show the process of automated overtaking at 4 different time instances: i) the third vehicle passing, ii) the first one taking the opposite lane, iii) the overtaking process, and iv) returning to the right lane. In this case, the third vehicle did not affect the overtaking process.

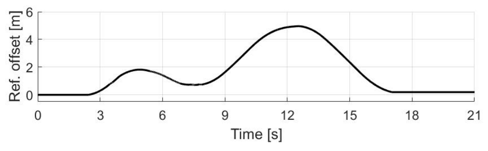
The information can be validated by observation of the control variables where Figure 6(e) shows the lateral offset planned for overtaking, without any change to the intention of the overtaking vehicle. It continues to increase the offset from 0 to 5 meters without any change. The value of 5 meters refers to the distance from the middle of the starting lane to the middle of the opposite lane.

Figure 6(f) and 6(g) depict good tracking of the lateral reference, with a lateral error e_{lat} of approximately $0.75[m]$ for around 3 seconds. The angular error e_{ang} (heading tracking error) has shown a maximum deviation between the vehicle heading and the angle of the reference path of 6° during the overtaking manoeuvre. Additionally, the smooth evolution of both variables results in a comfortable ride for the passengers. The speed remains constant after arriving at top speed, since the manoeuvre is not affected by the oncoming vehicle.

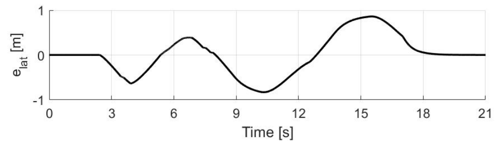
The second test case uses the same 3 vehicles. The sequence in Figure 7(a) shows a 200



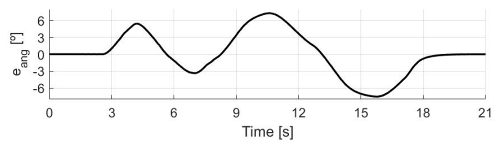
(a) Time secuence for overtaking in test case 2.



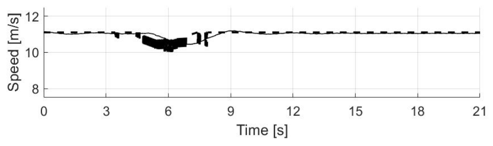
(b) Lateral offset.



(c) Lateral error.



(d) Angular error.



(e) Vehicle and reference speed.

Figure 7: Results of test case 2.

meter long road segment with two lanes, each 5 meters wide. In this test case, vehicle 1 is trying to execute an overtaking manoeuvre based on the information arriving from vehicle 2 and the distance between it and vehicle 3 (Figure 7(a) top part at 2 seconds). Vehicle 1 starts the overtaking manoeuvre, but the information sent by vehicle 3 causes vehicle 1 to remain in its lane and to adapt its speed (plots between 5 and 9 seconds). Then, vehicle 1 continues the overtaking manoeuvre (from 9 to 13 seconds) until it starts the returning process (from 13 to 17 seconds), arriving in the original lane at the end (19 seconds).

Figure 7(b) shows the process of returning to the right lane due to the blockage of vehicle 3 travelling in the opposite direction. It occurs between 5 and 9 seconds and it shows a reduction of the offset from 2 meters to less than 0.75 meters, doing a safe correction to avoid a possible collision.

Figures 7(c) and 7(d) show the tracking of the lateral controller, where the maximum lateral error e_{lat} is under 0.75 meters. The angular error e_{ang} is lower than 7 degrees with respect to the angle along the path.

Figure 7(e) depicts the speed correction made by the system to avoid the third vehicle for

Figure 8: DLR's automated vehicle FASCarE.

safety reasons. Vehicle 1 reduces the speed to adapt it to the speed of vehicle 2, avoiding a potential rear end collision.

2.4 Real vehicle results

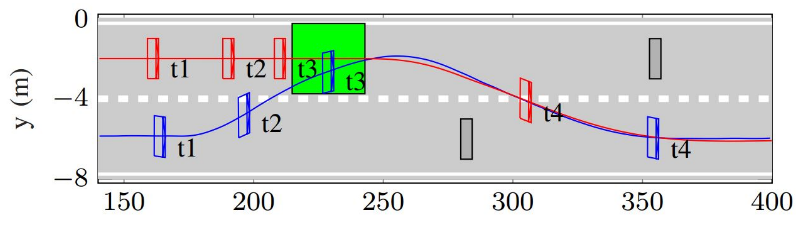
2.4.1 Scenario and tools

For this use case, DLR's vehicle (FASCarE) shown in Figure 8, was used. Three experiments have been carried out on the real platform plus a virtual automated vehicle and simulated obstacles. Experiment A and B use the same scenario of a 500m long two-lane road, with the same driving direction for both lanes and two static obstacles at distance of 280m and 350m. Both vehicles start at 20m with $v_x = 0$ in adjacent lanes and accelerate up to the maximum velocity of $16.7 m/s$.

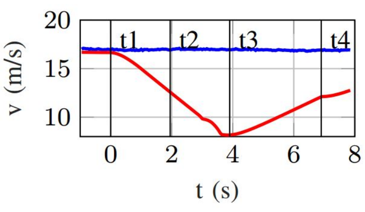
Experiment C is a simulated urban scenario with high lane curvatures and a varying number of lanes. As in A and B, two static obstacles are placed at different positions to perform the lane changes. A slow vehicle, with a maximum velocity of $7 m/s$ is placed 35 m ahead of the fast vehicle, which leads to the fast vehicle catching up to the slow vehicle right in front of the first blocking obstacle.

2.4.2 Results

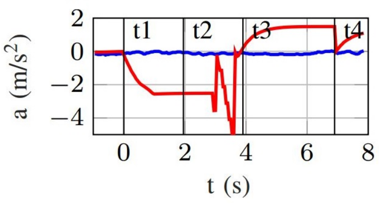
The results of experiment A are shown in Figure 9 for four points in time: before, during and after the cooperation. The physical vehicle 0 on the right lane and the simulated vehicle 1 on the left lane arrive at their top speed with only a minor deviation in x-distance. At t_1 vehicle 0 (blue) sends a cooperation request to vehicle 1 (red) to reserve the green area on the left lane for a reservation start time t_3 and directly receives a confirmation message. Vehicle 1 starts braking at t_1 to avoid the reserved area, so that vehicle 0 is ahead of vehicle 1 at t_2.



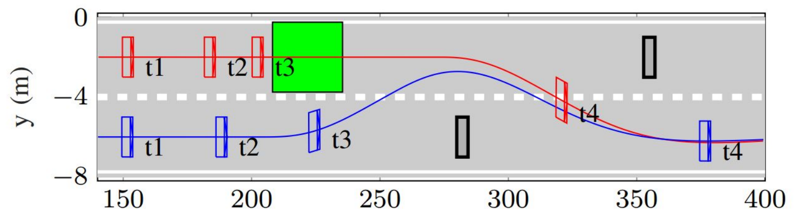
(a) Exp. A: trajectories and reservation.



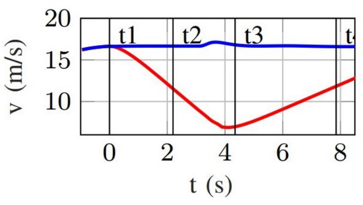
(b) Exp. A: velocity.



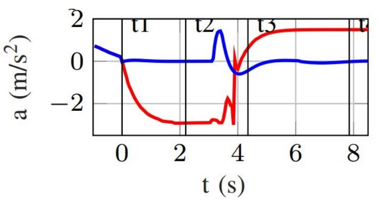
(c) Exp. A: acceleration.



(d) Exp. B: trajectories and reservation.



(e) Exp. B: velocity.

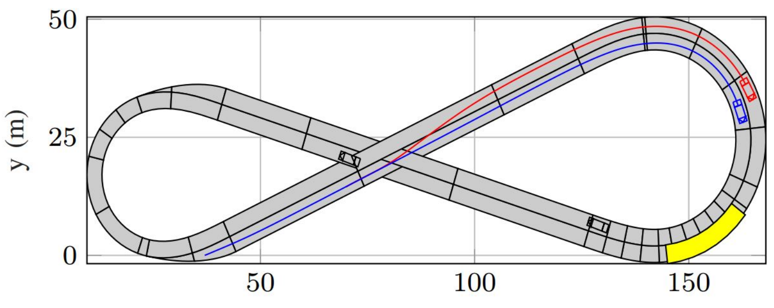


(f) Exp. B: acceleration.

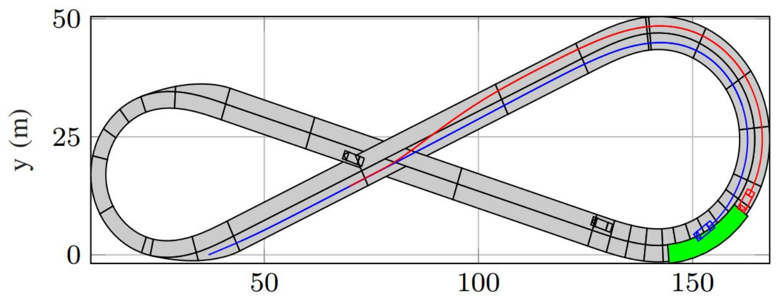
Figure 9: Cooperation of a physical and a simulated vehicle.

Vehicle 0 changes the lanes and enters the reserved area in front of vehicle 1. At t_3, vehicle 1 starts to accelerate back to the allowed speed limit. Both vehicles then non-cooperatively execute a lane change to the right lane, to avoid the second obstacle.

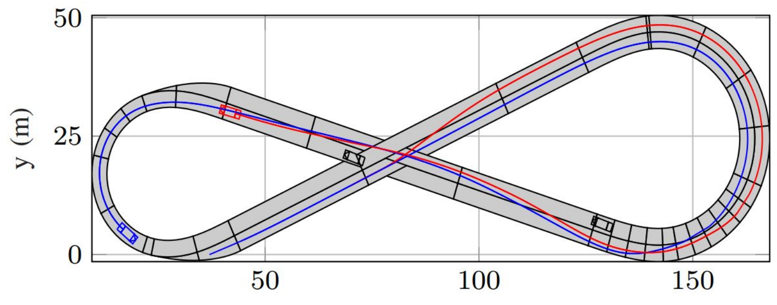
The results of experiment C are shown in Figure 10. The slower vehicle 1 (red) initially changes to the left lane at the entrance to the curve, as it is already aware of the blocked left lane at the exit of the curve. Vehicle 0 (blue) intends to drive faster and tries to overtake vehicle 1 by pursuing the inner, right lane. Due to the upper bound on lateral acceleration, it is limited to a similarly low speed inside the curve. At t = 80s depicted in Figure 10(a), vehicle 0 reserves a portion of the outer lane in front of vehicle 1. The reservation is shown in yellow, as it subsequently becomes active at t = 83s. Vehicle 1 reduces its velocity (coming to



(a) Taking the other lane.



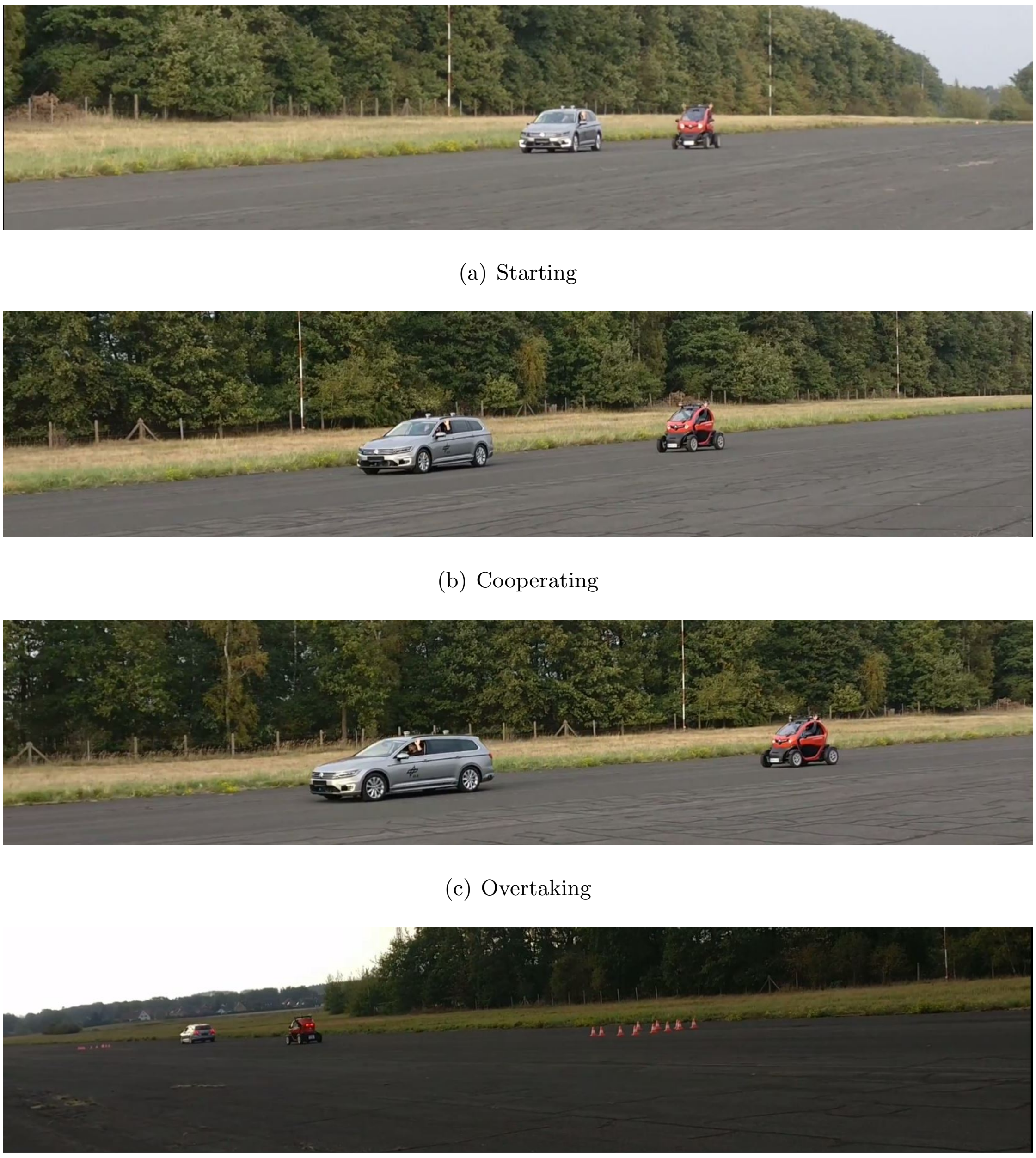
(b) Taking the other lane.



(c) Taking the other lane.

Figure 10: Results of test case 1.

a standstill at t = 83s due to the low initial velocity), in order to avoid the reserved area. As shown in Figure 10(b) for t = 83s, vehicle 0 changes lanes into the centre of the active (green) reservation area. With the faster vehicle 0 ahead, both vehicles go on to non-cooperatively change lanes to avoid the second obstacle on the left lane as well as the closure of the right lane (Figure 10(c) for t = 107s), while approaching admissible top speeds on the straight parts of the road and reducing the speed appropriately during curves.



(a) Starting

(b) Cooperating

(c) Overtaking

(d) Avoiding and finishing

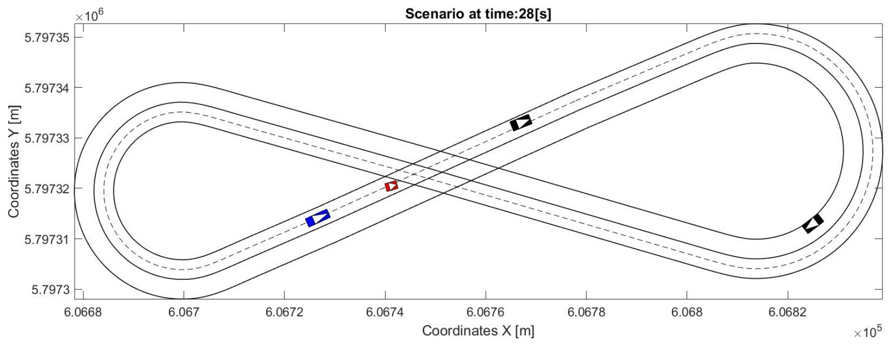
Figure 11: Cooperative overtaking between Tecnalia and DLR.

2.5 Real vehicle in cooperative scenarios

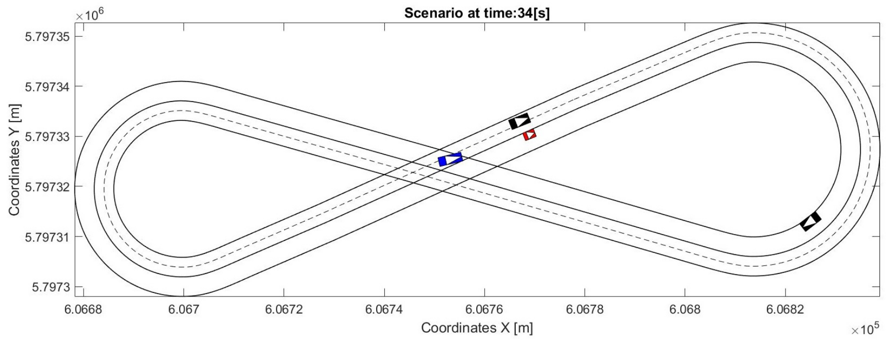
2.5.1 Scenario and tools

The final test case was done with both vehicles (DLR and Tecnalia) in a straight and curved environment. Both vehicles were running different programs and architectures, but use the same principles for trajectory planning and cooperation agreement. A sequence of pictures is shown in Figure 11.

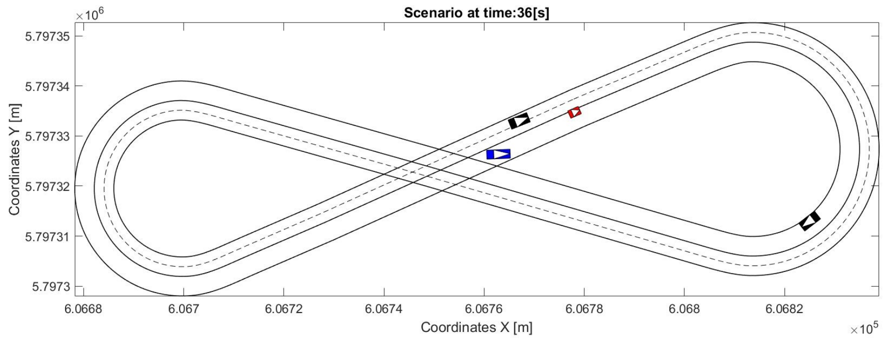
2.5.2 Results



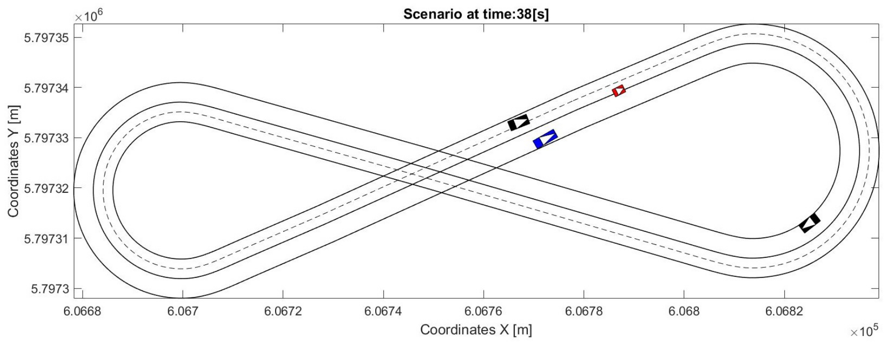
(a) Preparing the overtaking



(b) Tecnalia's overtaking process

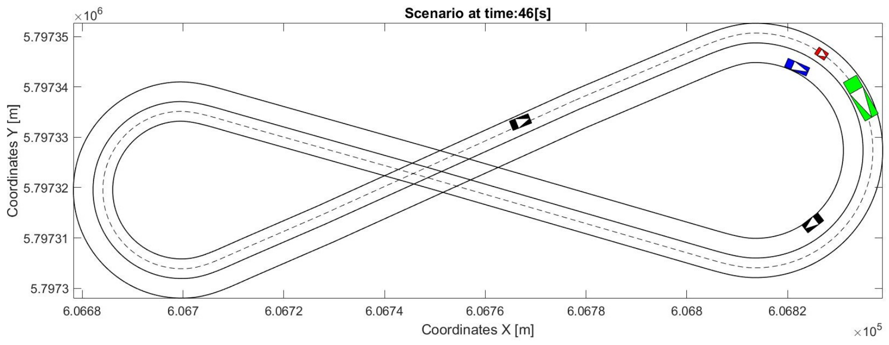


(c) DLR's overtaking process

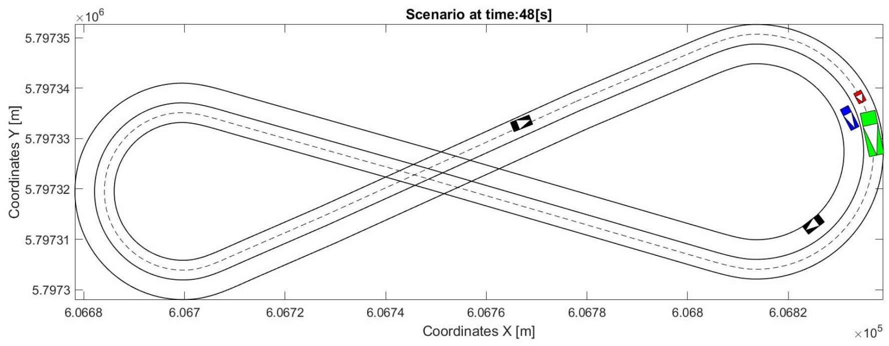


(d) Returning to the main lane

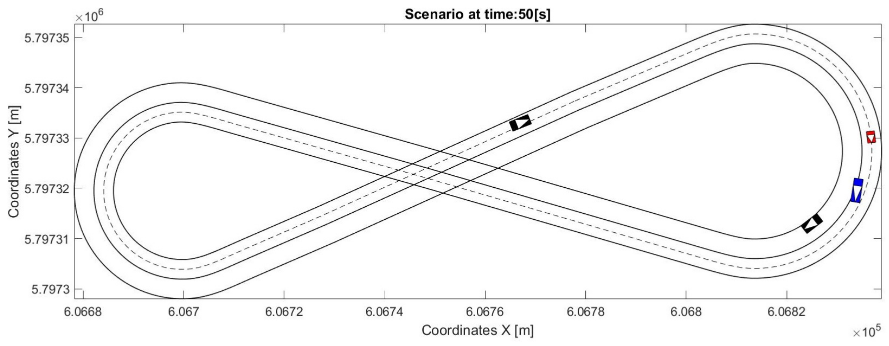
Figure 12: Overtaking scenario using Tecnalia and DLR platforms.



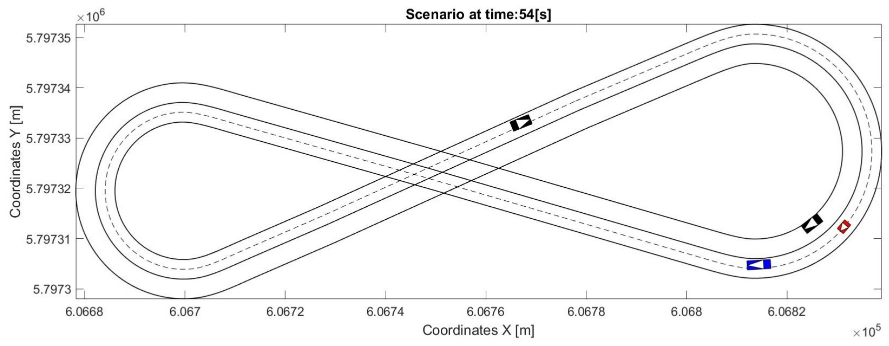
(a) Preparation of the cooperation agreement



(b) Acceptance of the cooperation



(c) Returning to the lane and cooperation finished



(d) Lane following in the urban scenario

Figure 13: Overtaking scenario using cooperation between Tecnalia and DLR platforms.

The results are summarized in Figures 12 and 13. In Figures 12 both vehicles did an overtaking manoeuvre considering a static obstacle. In this case the cooperation manoeuvre was not established because the lead vehicle was blocking the gaps for the follower. After this manoeuvre ends, the follower vehicle keeps to the right lane to analyse the possibility of a cooperative overtaking manoeuvre with the other automated vehicle (Figure 13). When the red vehicle accepts the cooperation agreement, it starts to consider the reservation as an obstacle that it must avoid, allowing the safe manoeuvring of the blue vehicle. After, the cooperation is finished the lane following process continues normally.

2.6 Conclusions

In the automated driving use case, a linear model predictive approach with a decoupled lateral and longitudinal model was developed and verified. This approach has the capability of managing standard manoeuvres for automated vehicles, as well as lane keeping, lane changing and overtaking. Additionally, a cooperation method is provided, based on communications between vehicles, optimizing the overtaking process.

The results show that introducing this linear model can achieve the safe future conditions considered in the manoeuvre planning. Additionally, it is fast enough to use other verification methods to activate external emergency manoeuvres in the worst case.

The cooperation agreements between vehicles have been done considering space reservations and kinematic models to project the future reservation space in time. This method shows good results in the case of complicated manoeuvres such as overtaking, improving and ensuring its safety.

3 Smart Grid Use Case

In this section we address an energy management problem for a smart grid according to a data-driven approach with probabilistic guarantees on the achieved performance.

We consider the smart grid testbed in Figure 14, which was set up at *General Electric Global Research Center (GERC)* in Munich, Germany, in the *Hertz Lab.* The testbed comprises only a few elements, that is photovoltaic (PV) solar panels, a (simulated) electrical load, a battery, and related interfaces to the control unit, with inverters and metering systems. The smart grid is connected with the main grid, which can eventually provide the energy needed for balancing demand and generation.

Electrical energy storage systems are becoming more and more widely used, in particular, as a means to deal with the intermittent and non-controllable nature of the power production by renewable energy supply systems such as solar PV installations and wind turbines, since they can serve the purpose of storing energy when production exceeds demand and releasing it when load request exceeds generation, see e.g. [14] and the references therein. Studies on electrical storage systems suitable for energy management in small grids or even single building applications, [31], and even stand-alone electricity generation systems relying to a significant extent on the supply of solar or wind energy, have been developed in the literature, [22].

The goal here is to define a strategy for the battery usage so as to compensate for the PV energy production fluctuations and guarantee that the energy exchange with the main grid is as close as possible to a nominal profile, within certified bounds, with reference to a one-day time horizon. This will reduce the uncertainty that the main grid is experiencing and, in a more global perspective of distributed energy resource systems, simplify the task of the main grid operator when defining the reservers and ancillary services needed to safely operate the grid, avoiding service disruption.

Due to the stochastic nature of the PV energy production, the problem is naturally formulated in a stochastic framework and the challenge becomes providing the probabilistic guarantees on the region where the energy request to the main grid is confined, based on a limited number of PV energy production profiles only, and some measurements of environmental data on irradiance and air temperature. Some energy production profiles of the PV installation and environmental data on irradiation and temperature are available for the design of the energy management strategy, together with a hybrid model for the battery and the electrical load profile. A data-driven solution resting on the so-called scenario theory, [13], originally introduced in [10, 11] to address robust design and then extended to chance constrained optimization in [12], is proposed.

It is worth noticing that, despite the simplicity of the considered set-up, the resulting smart grid energy management problem presents some of the key difficulties (control of a hybrid system modeling the battery, presence of stochastic uncertainty due to renewable energy sources) encountered in the energy management of large-scale smart grids.

All available data are described at the end of Section 3.3.2 and can be downloaded from the website `http://cps-vo.org/group/ARCH/benchmarks`.

The rest of the section is taken from [21] and organized as follows. We describe the smart grid configuration in Section 3.1. The benchmark energy management is presented in Section 3.2. A data-driven solution is proposed in Section 3.3 and its performance is assessed in Section 3.4. Concluding remarks are drawn in Section 3.5.

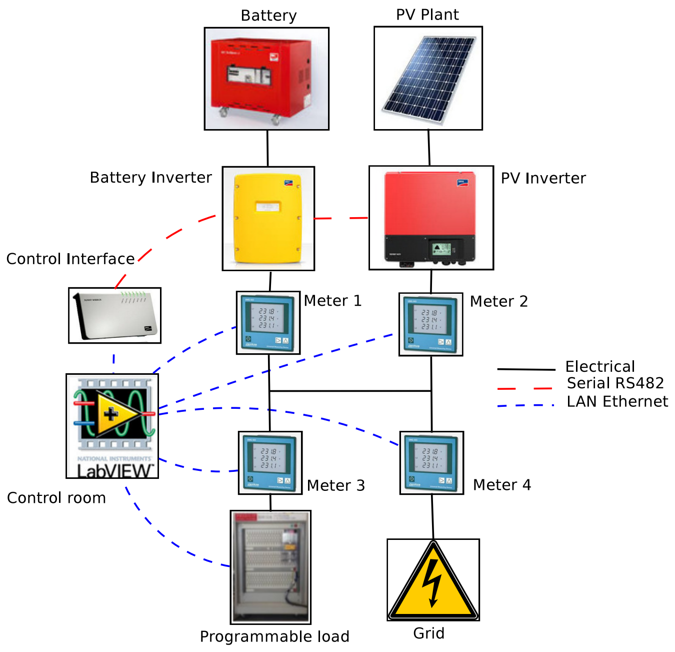


Figure 14: Smart grid configuration.

3.1 Description of the smart grid configuration

The testbed consists of a PV installation with the associated PV inverter, a programmable load simulator, a battery storage unit with its inverter, and a grid interface. Four smart meters provide measures of the involved energy flows, with the meters of the PV installation and a battery located downstream the corresponding inverters so as to include power conversion losses. A control interface is connected to the inverters via a serial RS-485 interface and can collect from them monitoring data and also set some of their configuration parameters. This latter functionality, in particular, allows to control the energy exchange with the battery.

The control strategy for the battery is implemented via NI Labview (`http://www.ni.com/labview/`), which runs on a desktop personal computer located in the control room. An Ethernet based Local Area Network (LAN) makes the control room communicate with the control interface, the meters, and the programmable load, via the MODBUS protocol.

Model of the electrical storage unit

A Li-ion battery from IBC-Solar is adopted for electrical energy storage in the considered testbed. The battery is combined with an SMA SunnyIsland inverter. A Battery Management System (BMS) at the interface of the battery with the inverter prevents its overheating (for safety) and deep discharge (to avoid shortening the battery life). The main role of the BMS is driving the inverter and selecting the working point on the voltage-current curve so as to track the desired power flow while pursuing the maximal efficiency for the battery.

From now on we will refer to the battery together with its inverter as the *storage*, and we shall derive a model of the storage for the design of the smart grid energy management strategy.

The plot on the left of Figure 15 represents the results of a power flow tracking experiment. Note that the power flow can be controlled with a high precision (of the order of 10 W) and that the storage system presents a fairly symmetric behavior when charging and discharging, with the power dynamics overshooting when the operational mode changes from charging to discharging or viceversa. Since the transient duration when commuting is of the order of 10 seconds and we are interested in a time scale of minutes, we shall neglect this fast dynamics in the model of the storage.

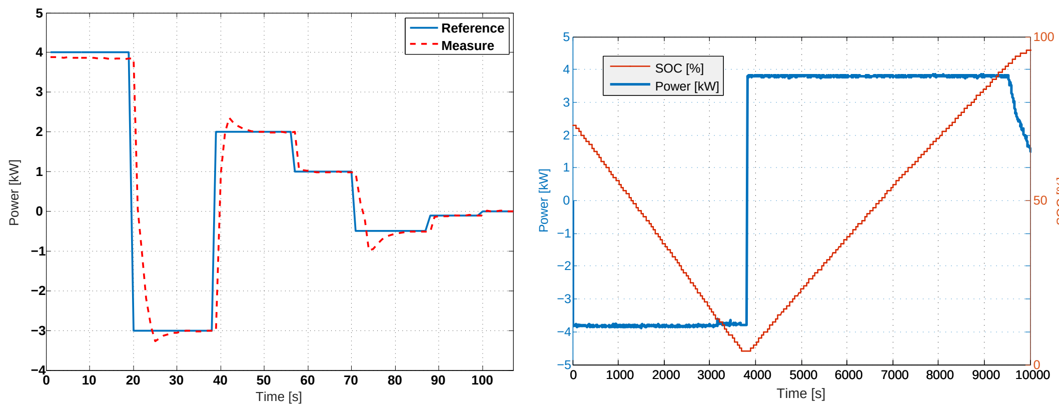


Figure 15: Power flow tracking experiment (left) and a cycle where the storage is discharged and charged at maximal rate (right).

Figure 15 shows on the right a cycle where the storage is first discharged and then charged at a maximum constant power rate of about ±3.8 kW when its state of charge (SOC) is between 5% and 95%. As the SOC grows beyond 95%, the maximum charging power rate decreases exponentially. Other experimental tests suggest not to reduce the SOC of the storage below 5% since the power flow control could be overtaken by the BMS to the purpose of preventing deep discharge. It is then convenient to limit the operational SOC range of the storage system between 5% and 95%.

The maximum power rates in charging and discharging that were tested are of about 3.797 kW and 3.822 kW, respectively. Charging and discharging rates depend on the SOC and the battery temperature. Empirical insight suggests to take 3.5 kW as maximum power rate in both the charging and discharging phases since this is the (maximum) value that is possible to track in all SOC and temperature operating conditions. The round trip efficiency is 4.07% and it is calculated as the energy mismatch for discharging the storage from 76% to 5% SOC and then recharging it back to 76% SOC as shown in the experiment reported on the right plot of Figure 15. The same experiment allows to estimate the capacity of the storage that is 24.780MJ (22.680 MJ according to the datasheet).

Based on the experimental observations described above, we can introduce a sampling time interval Δ and model the storage system as a discrete time hybrid system where the stored energy x evolves according to

$$x(k+1) = ax(k) + b(u(k))u(k), \tag{3}$$

where $u(k)$ represents the energy fed into ($u(k) > 0$) or drawn from ($u(k) \leq 0$) the storage in the k-th time slot of duration Δ, and it is re-scaled by coefficient

$$b(u) = \begin{cases} 1 - \varepsilon_u & u > 0 \\ 1 + \varepsilon_u & u \leq 0, \end{cases}$$

with $\varepsilon_u \in (0, 1)$. This coefficient accounts for the round trip efficiency: if we feed the storage with an amount of energy E_{in}, then the storage content increases by $(1 - \varepsilon_u)E_{in}$, and if we draw an amount of energy E_{out}, the storage content decreases by $(1 + \varepsilon_u)E_{out}$. Given that in a cycle where the storage is charged and discharged we waste a percentage of energy equal to 4.07%, then we can easily compute ε_u as follows $0.0407E_{in} = E_{in} - (1 - \varepsilon_u)E_{in}/(1 + \varepsilon_u)$, thus getting $\varepsilon_u \simeq 0.02$. Notice that we assume exchange losses to be symmetric between charging and discharging. Parameter a in (3) depends on Δ and accounts for the self discharging of the storage (Faradaic efficiency), which in the considered storage system turns out to be quite small. More precisely, if we set the sampling time equal to $\Delta = 1$ minute, then $a = 0.9998$.

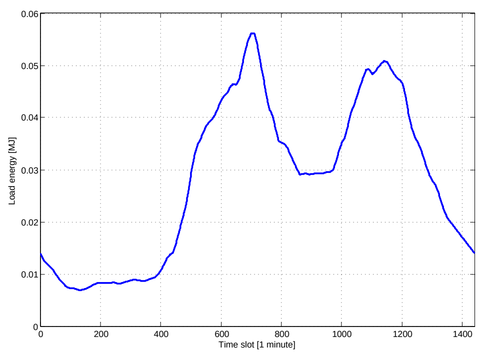


Figure 16: Simulated load profile. Values refer to electrical energy request per minute.

Model (3) is valid when the SOC of the storage is within 5% and 95%, which translates into the constraint

$$x_{\min} \leq x(k+1) \leq x_{\max}, \ k = 0, 1, \ldots, k_f - 1, \tag{4}$$

where $x_{\min} = 1.239$ MJ (5% SOC) and $x_{\max} = 23.54$ MJ (95% SOC). The energy exchanged with the storage is subject to the following constraint:

$$u_{\min} \leq u(k) \leq u_{\max}, \ k = 0, 1, \ldots, k_f - 1, \tag{5}$$

where $u_{\max} = -u_{\min} = \bar{u}\Delta$, $\bar{u} = 3.5$ kW being the maximum power rate in both the charging and discharging phases. If $\Delta = 1$ minute, then, $u_{\max} = -u_{\min} = 0.21$ MJ.

Electrical load

Prodigit 32612A is a device which can simulate both AC and DC electrical loads, with a maximum voltage of 300 V and maximum power of 5.4 kW, and is commonly used for inverter testing. In the smart grid testbed, it simulates the purely resistive load in Figure 16 that represents a typical household application rescaled so that its daily integral is comparable with the PV installation average daily energy production.

Notice that the simulated load is deterministic. The stochastic ingredient is brought into the problem by the PV energy production which is added to the load demand in the grid energy balance equation. The proposed approach to the design of an energy management strategy can be easily extended to the case when both the PV energy production and the load are stochastic.

PV installation & inverter

The testbed is equipped with a PV installation composed of 10 Hanwha HSL72 Polycristalline silicon modules 300 W each for an overall nominal power of 3 kW. These panels are placed

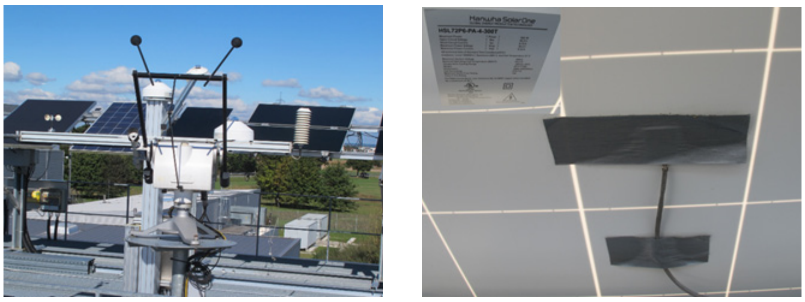


Figure 17: Weather station (on the left) and module temperature sensor (on the right).

on the roof of the *Hertz lab* and connected to the smart grid via a SMA Sunny Boy Inverter. The rack is facing south and is tilted by 28 degrees.

The SMA Sunny Boy converts the direct current of the PV array to grid-compliant alternating current and feeds it into the utility grid. The inverter can communicate with the control interface via a serial RS-485 interface.

A weather station equipped with irradiance and temperature sensors is placed on the roof of the *Hertz lab* (see Figure 17). Tilted global irradiance measures are made available through an irradiance silicon sensor tilted by 28 degrees and oriented to the south so that its measures are consistent with the radiation received by the array of PV panels. The temperature of each PV module is measured by a thermocouple placed on its backside, as shown in Figure 17. Measures are sampled every thirty seconds and stored in a database.

3.2 Energy management problem

Consider a one-day time horizon, discretized into $k_f = 1440$ time steps of length $\Delta = 1$ minute. The energy g exchanged with the main grid at step $k \in \{0, 1, \dots k_f - 1\}$ corresponding to the time frame $[k\Delta, (k+1)\Delta)$ is given by the energy balance equation

$$g(k) = \ell(k) - d(k) + u(k), \ k = 0, 1, \dots, k_f - 1, \tag{6}$$

where $g(k) > 0$ if the main grid provides energy to the smart grid in the k-th time frame, and $g(k) < 0$ viceversa. All quantities $\ell(k)$, $d(k)$, $u(k)$ refer to the k-th time frame and represent, respectively, the electrical load energy request, the energy produced by the PV installation, and the energy exchanged with the storage.

The load request ℓ is set equal to the (deterministic) profile in Figure 16. The energy exchanged with the storage u affects the storage content according to the model (3) in Section 3.1, and is subject to the constraints (4) and (5). The PV energy production profiles are uncertain and we assume that they are realizations of an underlying stochastic process, distributed according to some probability measure $\mathbb{P}$. Probability $\mathbb{P}$ is not known explicitly but only indirectly through a dataset of PV energy production profiles.

The goal is to appropriately set the charge/discharge of the battery so as to keep the energy exchange with the main grid along the one-day time horizon as close as possible to a suitably designed nominal profile, avoiding high fluctuations due to the energy production from renewable solar power. The nominal profile can be optimized so as to minimize the variability range of the energy exchanged with the main grid, while providing certified bounds on its extent. Given that the PV energy production is uncertain, the bounds are guaranteed in probability.

More precisely, let $\bar{g}(k)$, $k = 0, \ldots, k_f - 1$, be the nominal profile to be jointly optimized with the energy exchanged with the battery $u(k)$, $k = 0, \ldots, k_f - 1$. Then, the energy management problem can be formulated as the following chance-constrained optimization program

$$\min_{\substack{h_g, \bar{g}(k), k=0,\ldots,k_f-1 \\ u(k), k=0,\ldots,k_f-1}} h_g \tag{7}$$

$$\text{subject to: } \mathbb{P}\Big\{ \Big| \sum_{k=jw+1}^{(j+1)w} \delta g(k) \Big| \leq h_g, \; j = 0, 1, \ldots, \lfloor k_f/w \rfloor - 1,$$
$$x_{\min} \leq x(k+1) \leq x_{\max}, u_{\min} \leq u(k) \leq u_{\max}, k = 0, \ldots, k_f - 1 \Big\} \geq 1 - \epsilon,$$

where the extent of the fluctuations $\delta g(k) = g(k) - \bar{g}(k)$, $k = 0, \ldots, k_f - 1$, integrated over time windows of length w, $1 \leq w \leq k_f$, is minimized through its upper bound h_g, while imposing the actuation constraints (4) and (5). Note that the minimization of h_g is performed jointly with the constraint enforcement over a set of realizations of the PV energy production of probability larger than or equal to $1 - \epsilon$, where the violation probability $\epsilon \in (0, 1)$ is a design parameter and modulates the reachability region extent: the smaller ϵ, the larger the reachability region. The presence of the time window length parameter w adds some flexibility to the problem: if $w = 1$, then, fluctuations over a time scale of length $\Delta = 1$ minute are minimized, whereas choosing $w > 1$ can accomodate for a larger time scale of operation at the main grid level.

The challenge is now to solve the chance-constrained optimization program (7) – which is a difficult problem on its own, [35, 36] – knowing $\mathbb{P}$ only indirectly, through a limited number of PV energy production profiles and some measurements of environmental data on irradiance and air temperature.

We shall next describe a data-driven solution to this problem. The proposed approach can also be applied to variants of the problem like, e.g., the case when the nominal profile is assigned and given by the main grid operator, or when both PV generation and load are stochastic and $\mathbb{P}$ represents their joint probability distribution.

3.3 Proposed data-driven solution

We describe first the control policy parametrization and then the adopted scenario solution.

3.3.1 Control policy parametrization

Given that the PV energy production d is available through measurements and the goal is to limit its impact on the energy exchange with the main grid, we shall adopt a disturbance compensation scheme where the control input u is chosen as

$$u(k) = \gamma_k + \sum_{j=0}^{k-1} \theta_{k,j}(d(j) - \bar{d}(j)), \tag{8}$$

$\bar{d}$ being the expected value of the PV energy production d. The parameters $\gamma_k \in \mathbb{R}$ and $\theta_{k,j} \in \mathbb{R}$ are decision variables to be optimized. Let

$$\mathbf{u} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(k_f - 1) \end{bmatrix} \quad \mathbf{d} = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(k_f - 1) \end{bmatrix} \quad \boldsymbol{\ell} = \begin{bmatrix} \ell(0) \\ \ell(1) \\ \vdots \\ \ell(k_f - 1) \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(k_f - 1) \end{bmatrix}$$

be the vectors collecting all relevant quantities to the energy balance equation (6) along the reference one-day time-horizon. Then, (6) can be rewritten in vectorial form as

$$\mathbf{g} = \boldsymbol{\ell} - \mathbf{d} + \mathbf{u} \tag{9}$$

and the following expression can be derived for $\mathbf{u}$ as a function of $\mathbf{d}$:

$$\mathbf{u} = \Gamma + \Theta(\mathbf{d} - \bar{\mathbf{d}}), \tag{10}$$

where Γ and Θ are given by

$$\Gamma = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{k_f-1} \end{bmatrix} \quad \Theta = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \theta_{1,0} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \theta_{k_f-1,0} & \cdots & \theta_{k_f-1,k_f-2} & 0 \end{bmatrix}.$$

By plugging (10) into (9) and reordering some terms, we obtain

$$\mathbf{g} = \boldsymbol{\ell} - \bar{\mathbf{d}} + \Gamma + (\Theta - I)(\mathbf{d} - \bar{\mathbf{d}}). \tag{11}$$

We can then set

$$\overline{\mathbf{g}} = \ell - \bar{\mathbf{d}} + \Gamma \tag{12}$$

$$\delta\mathbf{g} = (\Theta - I)(\mathbf{d} - \bar{\mathbf{d}}) \tag{13}$$

and rewrite (11) as $\mathbf{g} = \overline{\mathbf{g}} + \delta\mathbf{g}$. The quantities $\overline{\mathbf{g}}$ and $\delta\mathbf{g}$ in (13) represent, respectively, the nominal exchange profile with the main grid and its fluctuations. Note that $\delta\mathbf{g}$ depends only on Θ and cannot be set equal to zero by imposing $\Theta = I$ since Θ is lower triangular due to the compensation scheme causality. Also, Θ cannot be set arbitrarily since the actuation constraints on the storage capacity and energy exchange rate (see (4) and (5)) have to be satisfied. The nominal profile $\overline{\mathbf{g}}$ depends only on Γ and not on Θ. However, the choice of Γ is coupled with that of Θ due to the actuation constraints, which are next written in vectorial form for uniformity. This is straightforward for the constraint on the energy exchange rate:

$$\mathbf{1}u_{\min} \leq \mathbf{u} \leq \mathbf{1}u_{\max}, \tag{14}$$

where $\mathbf{1}$ is a vector with all elements equal to 1 of the same dimension of $\mathbf{u}$. As for the storage capacity constraint, in order to make it explicit as a function of $\mathbf{u}$, we first need to define the stored energy vector $\mathbf{x} = \begin{bmatrix} x(1) & x(2) & \dots & x(k_f) \end{bmatrix}^\top$. Then, it is easily seen from (3) that

$$\mathbf{x} = Fx_0 + G\left(I - \varepsilon_u \mathrm{diag}(\mathrm{sign}(\mathbf{u}))\right)\mathbf{u} \tag{15}$$

where x_0 is the initial storage content, diag(sign($\mathbf{u}$)) is the diagonal matrix with the sign of the elements of $\mathbf{u}$ on the diagonal, and matrices F and G are given by

$$F = \begin{bmatrix} a \\ a^2 \\ \vdots \\ a^{k_f} \end{bmatrix} \qquad G = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a^{k_f-1} & \cdots & a & 1 \end{bmatrix}.$$

This finally leads to the following vectorial form for (4)

$$\mathbf{1}x_{\min} \leq Fx_0 + G\left(I - \varepsilon_u \mathrm{diag}(\mathrm{sign}(\mathbf{u}))\right)\mathbf{u} \leq \mathbf{1}x_{\max},$$

which – by observing that the inequality $|G\varepsilon_u \mathrm{diag}(\mathrm{sign}(\mathbf{u}))\mathbf{u}| \leq \varepsilon_u G\mathbf{1}\max_{k=0,1,\dots,k_f-1}|u_k|$ holds componentwise – can be enforced by adopting the conservative approximation

$$\mathbf{1}x_{\min} + \Xi h(\mathbf{u}) \leq Fx_0 + G\mathbf{u} \leq \mathbf{1}x_{\max} - \Xi h(\mathbf{u}), \tag{16}$$

where $\Xi = \varepsilon_u G\mathbf{1}$ and $h(\mathbf{u}) = \max_{k=0,1,\dots,k_f-1}|u_k|$. This approximation in practice reduces the model of the storage from hybrid to linear, and simplifies the design of the energy management strategy.

We can now finally formulate the energy management problem in terms of the following chance-constrained optimization program

$$\min_{\Gamma,\Theta,h_g,h_u} h_g + \rho_g \parallel \bar{\mathbf{g}} \parallel_2 + \rho_u h_u \tag{17}$$

$$\text{subject to:} \qquad \mathbb{P}\Big\{\Big|\sum_{k=jw+1}^{(j+1)w} \delta g(k)\Big| \leq h_g,\ j=0,1,\dots,\lfloor k_f/w \rfloor - 1, |\mathbf{u}| \leq \mathbf{1}h_u,$$
$$\mathbf{1}x_{\min} + \Xi h_u \leq Fx_0 + G\mathbf{u} \leq \mathbf{1}x_{\max} - \Xi h_u, \mathbf{1}u_{\min} \leq \mathbf{u} \leq \mathbf{1}u_{\max}\Big\} \geq 1-\epsilon$$

which differs from (7) in that the penalization terms $\rho_g \parallel \bar{\mathbf{g}} \parallel_2$ and $\rho_u h_u$ with weights $\rho_g, \rho_u > 0$ are added to h_g. This is in order to have a uniquely defined solution for Γ, and to reduce the conservatism of the approximated storage capacity constraint (16), respectively.

3.3.2 Scenario solution

Given that the probability $\mathbb{P}$ is not known and only a set of realizations $\{\mathbf{d}^{(i)},\ i=1,\dots,N\}$ extracted from $\mathbb{P}$ is available, we shall formulate the following scenario optimization problem with constraint removal to approximate the original chance-constrained program:

$$\min_{\Gamma,\Theta,h_g,h_u} h_g + \rho_g \parallel \ell - \bar{\mathbf{d}} + \Gamma \parallel_2 + \rho_u h_u \tag{18}$$

$$\text{subject to:} \quad \begin{cases} \Big|\sum_{k=jw+1}^{(j+1)w} (\Theta - I)(\mathbf{d}^{(i)}(k) - \bar{\mathbf{d}}(k))\Big| \leq h_g,\ j=0,1,\dots,\lfloor k_f/w \rfloor - 1 \\ \mathbf{1}x_{\min} + \Xi h_u \leq Fx_0 + G\Gamma + \Theta(\mathbf{d}^{(i)} - \bar{\mathbf{d}}) \leq \mathbf{1}x_{\max} - \Xi h_u \\ \mathbf{1}u_{\min} \leq \Gamma + \Theta(\mathbf{d}^{(i)} - \bar{\mathbf{d}}) \leq \mathbf{1}u_{\max} \\ |\Gamma + \Theta(\mathbf{d}^{(i)} - \bar{\mathbf{d}})| \leq \mathbf{1}h_u \end{cases}$$
$$i \in \{i_1, i_2, \dots, i_{N-\lfloor \eta N \rfloor}\},$$

where $\eta \in [0,\epsilon)$ is the empirical violation parameter and $\{i_1, i_2, i_{N-\lfloor \eta N \rfloor}\} \subseteq \{1,2,\dots,N\}$ contains the indices of $N - \lfloor \eta N \rfloor$ out of the N available PV energy production realizations, which should be selected so as to best improve the cost. Each one of the removed $\lfloor \eta N \rfloor$ realizations should change the solution if included in the scenario program (18).

If N is appropriately selected, the solution to the scenario program (18) is feasible for the original chance constrained problem (17). This is stated in the following proposition whose proof follows directly from [12] with the only difference that the cost is not linear and one has then to apply the considerations stated in the proof in [16, Appendix A].

Proposition 1. *Select a confidence parameter $\beta \in (0,1)$ and an empirical violation parameter*

$\eta \in [0, \epsilon)$. *If* N *satisfies*

$$\binom{\lfloor \eta N \rfloor + n - 1}{\lfloor \eta N \rfloor} \sum_{i=0}^{\lfloor \eta N \rfloor + n - 1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \leq \beta,$$

where n *denotes the number of optimization variables in* (18)*, with probability no smaller than* $1-\beta$*, the solution to the scenario optimization problem with constraint removal* (18) *satisfies the probabilistic constraint in* (17)*.*

Not surprisingly, feasibility of the scenario solution holds with a certain confidence $1-\beta$. This is because the scenario solution is a random quantity that depends on the extracted multi-sample $d^{(1)}, d^{(2)}, \ldots, d^{(N)}$. If a bad multi-sample is extracted, the feasibility property does not hold, but this event becomes more and more unlikely as N increases. The dependence on β is logarithmic (see [5, 34]) so that it can be set quite small without growing too much N. Though the feasibility of the randomized solution is guaranteed for every $\eta \in [0, \varepsilon)$, the closer η is to the desired violation probability ε, the better the randomized solution approximates the actual solution to the chance-constrained problem. At the same time, however, N grows to infinity as $O(\frac{1}{\varepsilon - \eta})$ when $\eta \to \varepsilon$, [12], so that one should choose η based on the available computational resources.

The proposed data-driven approach requires a relatively large collection of electrical energy production profiles from the PV installation in the smart grid. 30 day of measurements were performed in August 2016. Only 28 out of the 30 PV power production profiles turned out to be not corrupted due to malfunctioning of the measurement system. PV module temperature, irradiation, and air temperature profiles were also collected. A further dataset of 220 irradiation and ambient temperature profiles is available. Data were collected during July, August, September of 2010, 2011, 2015, and 2016. The idea is then to reconstruct further PV power production profiles from this latter dataset. This is described in the following Section 3.3.3.

3.3.3 PV energy production profiles reconstruction

Reliable profiles can be reconstructed from environmental data on air temperature and irradiance. This topic is well investigated in literature (see e.g. [37]) and the use of photovoltaic power models is common practice in applied power forecasting, plant performance evaluation and design [15]. The module power production $P_M(t)$ at time t can be expressed as a static function of the irradiance on the tilted surface $I(t)$ and the temperature of the module $T_M(t)$ as

$$P_M(t) = \overline{P} \frac{I(t)}{\overline{I}} \Big(1 - \frac{\alpha}{100} \big(T_M(t) - \overline{T}\big)\Big), \tag{19}$$

where $\overline{T} = 25^o$C and $\overline{I} = 1$ kW/m^2 are the standard test temperature and irradiance, $\overline{P}$ represents the nominal power production of the panel, measured in the testing conditions of $T_M = \overline{T}$ and $I = \overline{I} = 1$ kW/m^2. The temperature coefficient α is introduced to take into account the performance degradation of the panel due to overheating. In normal working conditions a module can heat up to 70/80^{o}C (how much depends on the panel technology and application), and this affects significantly the solar cell efficiency [17]. This is the case, in particular, for the polycrystalline solar panels in the testbed, which have slightly higher temperature coefficients than panels built with other technologies (e.g. Cadmium Telluride CdTe solar panels).

We shall next use the dataset of August 2016 to determine the parameters in model (19). We shall then identify a model that provides the temperature of the module, based on the irradiance and air temperature. By means of the two identified models we are able to reconstruct the PV energy production profiles from the dataset containing only irradiance and air temperature profiles.

Power model identification: Equation (19) can be rewritten in regression form as

$$P_M(t) = \nu^\top \phi(t), \tag{20}$$

where $\nu^\top = \begin{bmatrix} \overline{P} & \alpha\overline{P} \end{bmatrix}$ and $\phi(t)^\top = \begin{bmatrix} I(k)/\overline{I} & -I(k)10^{-2}/\overline{I}\big(T_M(t) - \overline{T}\big) \end{bmatrix}$.

Data of PV power production, irradiance, and module temperature measured over 30 days in August 2016 are reported in Figure 18, with a sampling time of 1 minute for the power production and 30 seconds for irradiance and temperature. Over 43200 collected samples, corrupted data due to inactivity of the test plant (i.e. maintenance) and/or missing data from the weather or temperature measurement device and night time values are discarded. $n_d = 19337$ samples, corresponding to 90% of the available data, are used to identify the parameters vector ν by the least squares method, i.e.,

$$\nu = \left(\sum_{k=1}^{n_d} \phi(k)\phi(k)^\top \right)^{-1} \sum_{k=1}^{n_d} \phi(k) P_M(k).$$

The obtained parameters are $\overline{P} = 2.1096$ kW and $\alpha = 0.6566^o$C-1, which are similar to the values in [38]. The remaining 10% of the data are used for model validation. Statistics of the power model error are presented in Figure 19. From the empirical cumulative distribution function of the absolute value of the power error, we can observe that with 90% probability it is lower that 0.1842 kW, which is less than 8% of the nominal power $\overline{P}$. The expected value for the power error is 0.0084 kW and from the histogram we can see that the error is quite concentrated around the mean.

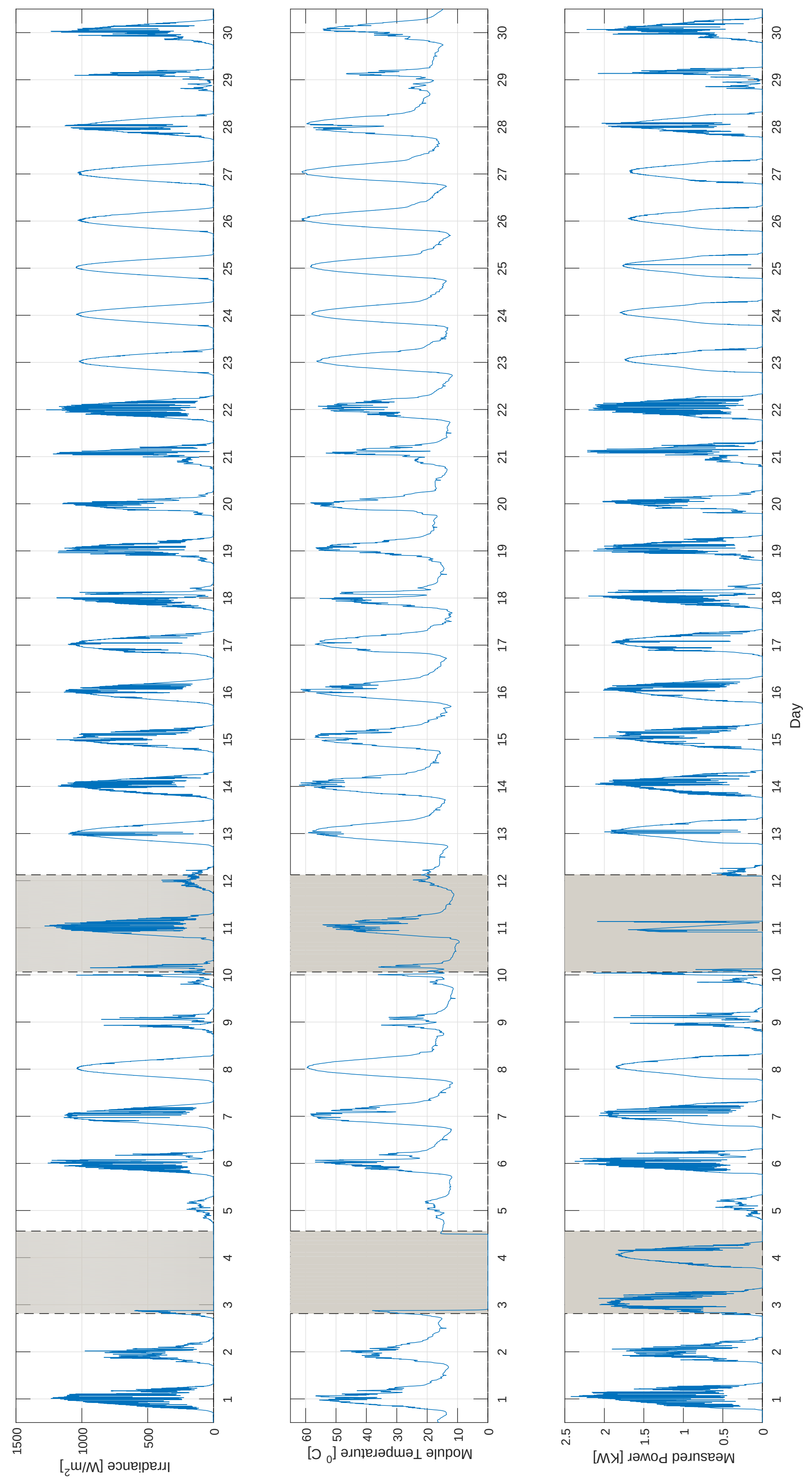


Figure 18: PV energy profiles and environmental data collected in August 2016. Gray areas correspond to days where some of the measurements were corrupted.

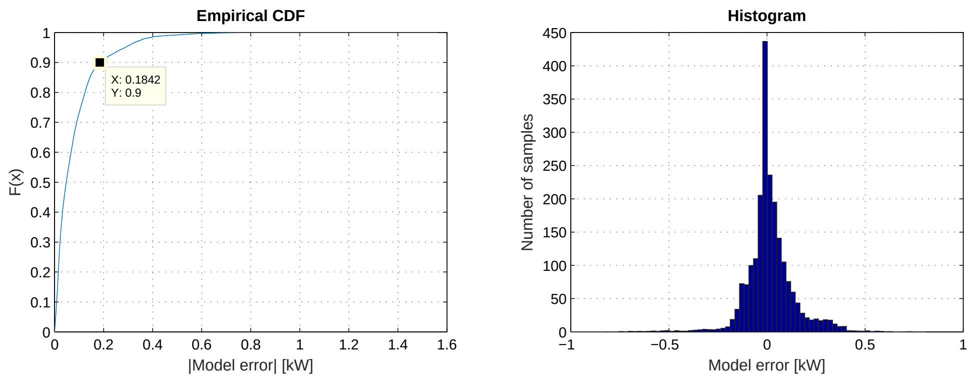


Figure 19: Power model error statistics.

Thermal model identification In order to reconstruct the module temperature from environmental measures of air temperature and irradiance, we propose a variant of the following model, that is proposed by Sandia National Laboratories, a multi-program laboratory managed and operated by Sandia Corporation operating for the U.S. Department of Energy's National Nuclear Security (`http://pvpmc.sandia.gov/modeling-steps/2-dc-module-iv/module-temperature/`):

$$T_M(t) = c_T I(t) e^{b_1 + b_2 W(t)} + T_A(t), \tag{21}$$

where $c_T = 1^o\text{CW}^{-1}\text{m}^2$, and b_1 and b_2 are parameters that depend on the module construction and materials and on its mounting configuration and their values can be found in technical references and manuals for various common module types and configurations. Model (21) is static and expresses the temperature of the module $T_M(t)$ at time t as a function of the air temperature $T_A(t)$, wind speed $W(t)$ and irradiance $I(t)$. It is conceived for the estimation of the average module temperature over relatively long time periods (i.e., with a duration larger than 10 minutes). In our case, we need a smaller time resolution and, hence, we introduce a first-order dynamics modeling the thermal transients of the module, thus replacing the static model (21) with the following discrete time dynamic model:

$$T_M(k) = a_1 T_M(k-1) + a_2 \Big(c_T I(k) e^{b_1 + b_2 \overline{W}} + T_A(k) \Big). \tag{22}$$

Note that in (22) the wind speed is replaced with its average value $\overline{W}$, which will then be estimated from data together with b_1 and b_2.

In order to retain the physical features of the original static model, parameters a_1 and a_2 are chosen so that in stationary conditions (i.e., when $T_M(k) = T_M(k-1)$) model (22) matches the original static model

$$\frac{a_2}{1-a_1} = 1 \Leftrightarrow a_2 = 1 - a_1. \tag{23}$$

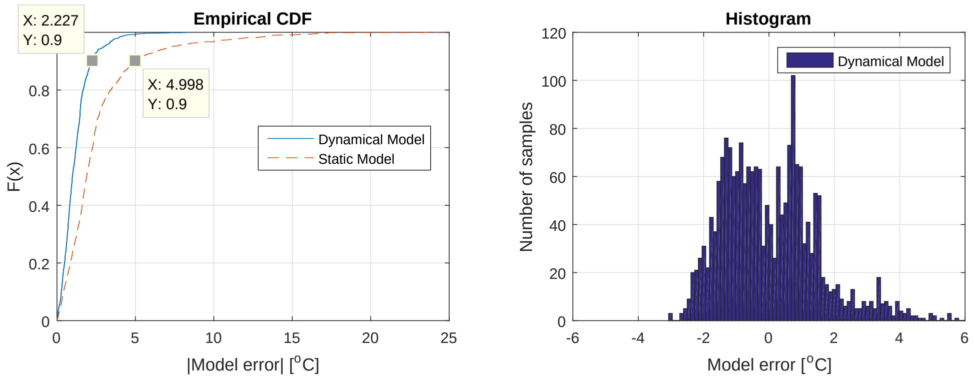


Figure 20: Module temperature error statistics.

Finally, by setting $a_3 = (1 - a_1)e^{b_1 + b_2\overline{W}}$ and replacing a_2 with its expression as a function of a_1 in (23), model (22) is re-parameterized as

$$T_M(k) = a_1 T_M(k-1) + a_3 c_T I(k) + (1 - a_1)T_A(k). \tag{24}$$

Parameters a_1 and a_3 are then identified by minimizing the mean square prediction error

$$J(a_1, a_3) = \frac{1}{n_d}\sum_{k=1}^{n_d}\Big(T_M(k) - a_1 T_M(k-1) + a_3 c_T I(k) + (1 - a_1)T_A(k)\Big)^2,$$

subject to the physical constraints $0 \leq a_1 < 1$ and $a_3 > 0$. Data referring to day time only were used in the identification since nocturnal radiative cooling effects are not captured in the proposed model. Identification is performed on $n_d = 5436$ samples, roughly 70% of daytime data measured every 30 seconds in a 6 days long record, the remaining 2300 (30% of data) are then used for validation.

The identified parameters values are $a_1 = 0.8682$ and $a_3 = 0.0046$. The mean value of the prediction error (considering day time only) is 0.049^oC. The empirical cumulative distribution function of the absolute value of the prediction error plotted in Figure 20 shows that for 90% of the samples the magnitude of the prediction error is smaller than 2.227 ^oC for the dynamical model (22). This value roughly doubles in the case we use the original static model. This error magnitude appears acceptable based on some studies in the literature (see e.g., [25]).

Power data reconstruction: We use irradiance and air temperature data records for the period of July, August, September of the years 2010, 2011, 2015, and 2016 for an overall number of 220 daily realizations to first reconstruct the module temperature and then to obtain realizations of PV power production. Some post processing to lower the noise on top of the tails for the reconstructed realizations due to the irradiance sensor was then performed. In Figure 21 the daily mean profile of the reconstructed data set and that of

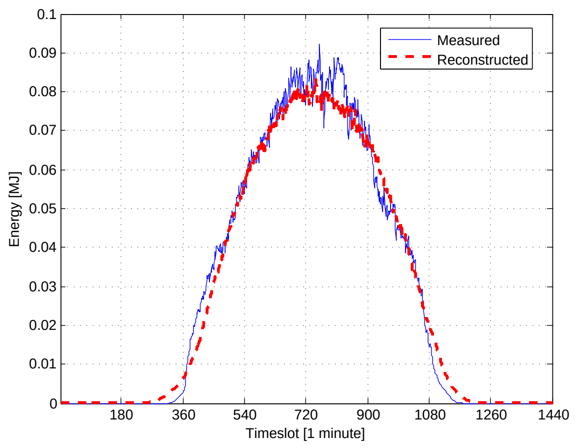


Figure 21: Comparison between empirical mean of the PV energy production profile per minute obtained based on the 28 measured realizations (blue line) and on the 220 reconstructed realizations (red line).

the real measurements are compared: the latter is more oscillating and noisy because of the limited number of realizations (only 28 measurements). The main differences can be found at the very beginning and at the end of the production period, which are most probably due to some model error for the power production at low value of irradiance.

3.4 Performance assessment of the proposed energy management strategy

We shall next illustrate the performance that one can achieve with the proposed approach, based on the available dataset.

Recall that the sampling time is set equal to $\Delta = 1$ minute. The storage system is initially charged at 50% SOC, i.e., $x_0 = 12.390$ MJ. Γ and Θ in (10) are parameterized as

$$\Gamma = \gamma \bar{d}, \qquad \Theta = \begin{bmatrix} 0 & 0 & \cdots & 0 & \\ \theta_1 & 0 & \ddots & \vdots & \\ \theta_2 & \theta_1 & 0 & & \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \cdots & \theta_2 & \theta_1 & 0 \end{bmatrix}$$

in order to reduce the number of optimization variables in the scenario optimization program (18) to $n = 5$. The weighting parameters in (18) are set equal to $\rho_g = \rho_u = 10^{-4}$. The time window length is $w = 10$. Confidence, violation, and empirical violation parameters are given by $\beta = 10^{-3}$, $\epsilon = 0.15$, and $\eta = 0.035$. Correspondingly, the bound in Proposition 1 is satisfied with $N = 220$ reconstructed scenarios of PV energy production. The number of constraints to be removed is 7. The expected value of the PV energy production $\bar{\mathbf{d}}$ was estimated as the empirical mean of the 220 reconstructed production profiles (see Figure 21).

The obtained solution of the scenario optimization program (18) is: $\gamma^\star = 0.1290$, $\theta_1^\star = 0.3495$, $\theta_2^\star = 0.1720$, $h_g^\star = 0.3324$, $h_u^\star = 0.0455$.

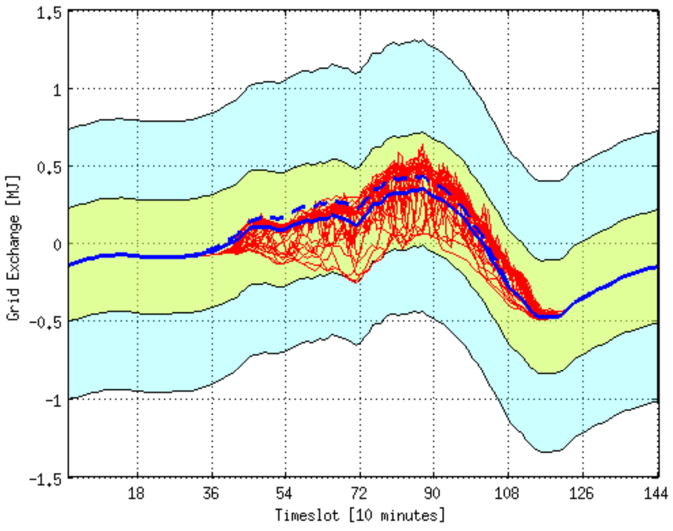


Figure 22: ϵ-reachability region obtained with the designed energy management strategy using the storage (green inner tube centered on the solid blue line) together with the energy exchange profiles with the main grid corresponding to the 28 measured PV production realizations (red lines). The ϵ-reachability region obtained without storage is the wider light blue outer tube centered on the dashed blue line.

The green inner tube centered on the nominal profile $\bar{\mathbf{g}}^\star = \gamma^\star \bar{\mathbf{d}} + \ell - \bar{\mathbf{d}}$ (blue solid line) with width $2h_g^\star$ in Figure 22 represents the ϵ-reachability region containing all profiles of energy exchange with the main grid integrated over time windows of length $w\Delta = 10$ minutes except for a set of probability smaller than $\epsilon = 0.15$, with high confidence ($\geq 1 - \beta = 1 - 10^{-3}$). The 28 realizations of energy exchange with the main grid corresponding to the measured PV energy production profiles are plotted in the same figure, for validation purposes. For 4 of them, the constraints in the optimization program are violated. This fraction is actually smaller than $\epsilon = 0.15$. In Figure 22, we also plot the ϵ-reachability region obtained by solving (18) with $\gamma = \theta_1 = \theta_2 = 0$, i.e, removing the storage system. In this case we obtain $h_g^\star = 0.87026$, which maps into the light blue outer tube centered on the dashed blue line representing the reference nominal profile $\bar{\mathbf{g}} = \ell - \bar{\mathbf{d}}$.

In Figure 23, we plot the SOC evolution obtained with the optimal energy management strategy evaluated on the 28 validation PV energy production profiles (blue lines). The red thick line is the nominal charging profile $\Gamma^\star = \gamma^\star \bar{\mathbf{d}}$. The green dot-dashed lines represent the conservative bounds adopted in (16). They are violated for 4 of the profiles. The actual

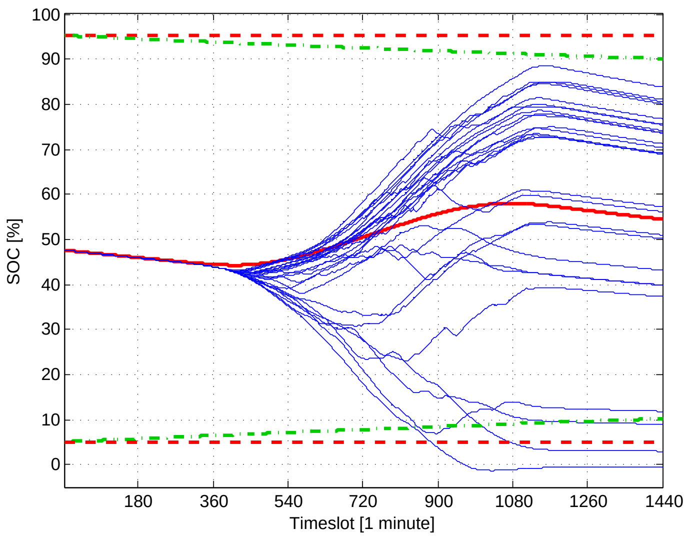


Figure 23: Energy exchange with the storage for the measured PV energy production profiles obtained with the designed energy management strategy (blue lines). The red solid line is its nominal component. The red dashed and greed dot-dashed lines represent, respectively, the actual storage capacity constraints and their conservative approximation adopted in the optimization program.

bounds (red dashed lines) are violated only for 2 of them. Note that the SOC evolutions reported in Figure 23 include also profiles where the battery capacity constraints are violated. This is possible because we are simulating the battery behavior for the purpose of verifying that the ϵ violation probability is not exceeded. When implementing the proposed strategy on the real battery, the control input will be set to zero (no energy exchange with the battery) when the actual bounds on its capacity are violated.
Alternative strategies can be compared in terms of width of the ϵ-reachability region for a given time window length w. Computation time is not an issue when the design is performed off-line as in our approach. It becomes a relevant criterion when adopting on-line design strategies like in model predictive control [30], where constrained optimization is repeated at each time step, based on the current measurements, and only the first control action is applied at each step (receding horizon implementation).

3.5 Concluding remarks

In this section, we consider energy management of a smart grid, where a storage system is used to compensate the fluctuations of solar PV energy production so as to provide a certified profile of the energy exchange with a main grid together with tolerance bounds.

The high penetration in the power grid of renewable energy sources characterized by an uncertain generation profile calls for the design of suitable strategies to guarantee the balance between consumption and generation in order to avoid voltage instabilities and possible outages. Solutions based on ancillary services and reserves are costly, and the idea developed here is that fluctuations in energy production from PV panel installations could be compensated at the level of the smart grid, thus alleviating the main grid operator task and reducing costs. This has potential for significant impact at societal level in supporting low-carbon economy and reducing carbon emissions.

The main challenge in addressing the problem is that only a limited number of data is available for the energy management strategy data-driven design. As discussed in [8], this highly constrains the complexity of the energy management strategy, which in our design was given by a disturbance compensator parameterized by only three scalar optimization variables.

Different control parameterizations and control strategies can be conceived. If a receding horizon implementation is adopted, then, improved predictions of the actual PV energy production profile could be obtained via filtering techniques and used for the on-line design of the control input, as suggested in [9].

A possibility to enable the design of a more complex and better performing energy management strategy is to build a stochastic model of the PV energy production from the available data, and then use this model to extract further realizations. This opens up a new topic on the prediction of solar PV power production and requires further investigation. A simple stochastic model based on principal components analysis was used in the recent UnCoVerCPS work in [18] to generate the solar PV power production profiles and derive via a scenario-based approach a periodic control strategy to optimally operate the battery in (cyclo)stationary conditions.

4 Human-robot collaboration use case

The GRAIL Robot System, shown in Figure 24, is the system around which the human-robot collaboration use case is based and is intended to be used for food assembly tasks carried out in collaboration with human co-workers and other robots. It uses a vision system to locate ingredients and bases (the bread in this case) and assembles the sandwiches according to an operator-defined recipe. The user interface allows the recipes to be created in a graphical drag and drop manner making it easy for even non-technical people to operate the system.

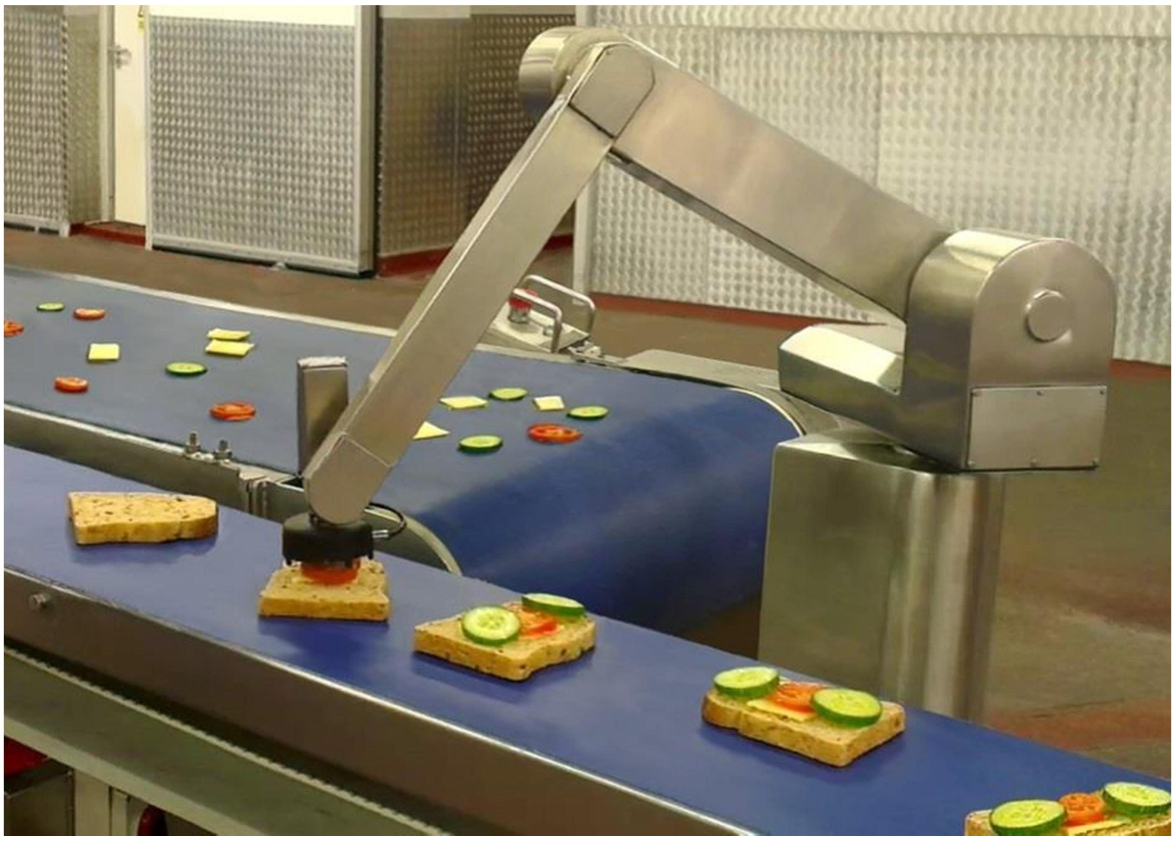


Figure 24: The GRAIL Robot System in operation.

Since most food manufacturing companies are SMEs, they often cannot afford the large investment necessary to completely automate a production line. To alleviate this barrier to adoption, the GRAIL Robot System is designed to be deployable alongside other workers (human or robotic) on an incremental basis. It consists of a low-cost, lightweight, hygienic robot arm along with camera systems (not shown), control system (not shown) and user interface (not shown).

The GRAIL Control System is used to control the robot arm so that it assembles sandwiches or other food products from the ingredients that arrive on one or more conveyors and bases that arrive on a different conveyor. It also has the goal of assembling these sandwiches as efficiently as possible so that fewer robots or human co-workers are required to meet the necessary throughput requirements.

The GRAIL Robot Control System consists of two parts: the motor control system and the GRAIL Control System. The motor control system is responsible for the low-level control of the motors in the robot joints and utilise, at the lowest level, commercially available motor controllers, one for each robot axis. These provide closed-loop position control of the robot joint positions using a PID control algorithm. The set points for these motor position control systems are provided by the GRAIL Control System. The motor control system also contains the actuator that controls the air supply to the gripper which is also controlled via the CAN bus interface.

The GRAIL Control System uses a vision system to determine the position of the ingredients and bases on the conveyors and generates the appropriate motion of the robot joints and actuation of the gripper to achieve the goal of assembling sandwiches. It also takes inputs from position encoders fitted to the conveyors so that the motion of the ingredients and bases along the conveyors can be accounted for.

The original GRAIL Robot System had no mechanisms to prevent collisions between the robot and human co-workers when they are working side-by-side on the same production line. The system could only be made safe by the provision of physical barriers between the robot and co-workers. In a typical factory, especially one used by an SME, space is limited and bulky cages or other physical separation methods are not appropriate.

Traditional alternative methods using light curtains or safety-rated vision systems are too detrimental to the throughput of the system since they normally operate the robot's emergency stop (E-Stop) circuit which requires a time-consuming restart procedure after activation.

4.1 The GRAIL Simulation

A comprehensive graphical simulation of the GRAIL robot system has been produced (see Figure 25). The purpose of this is twofold: first, it allows convenient debugging of the high-level robot control code without the difficulty and risks associated with testing new software on the real robot system, and secondly it allows for the efficiency of the system, in terms of the rate at which it can assemble sandwiches, to be measured under various scenarios without the time-consuming job of proving enough ingredients and bases on the conveyors to keep the system busy. The simulation also makes it possible to generate more precisely controllable and repeatable system inputs in the form of ingredient arrival rate, bases arrival rate and co-worker behaviour than would be possible using the real physical system. This makes it possible to extract more meaningful numerical data from the tests that are performed.

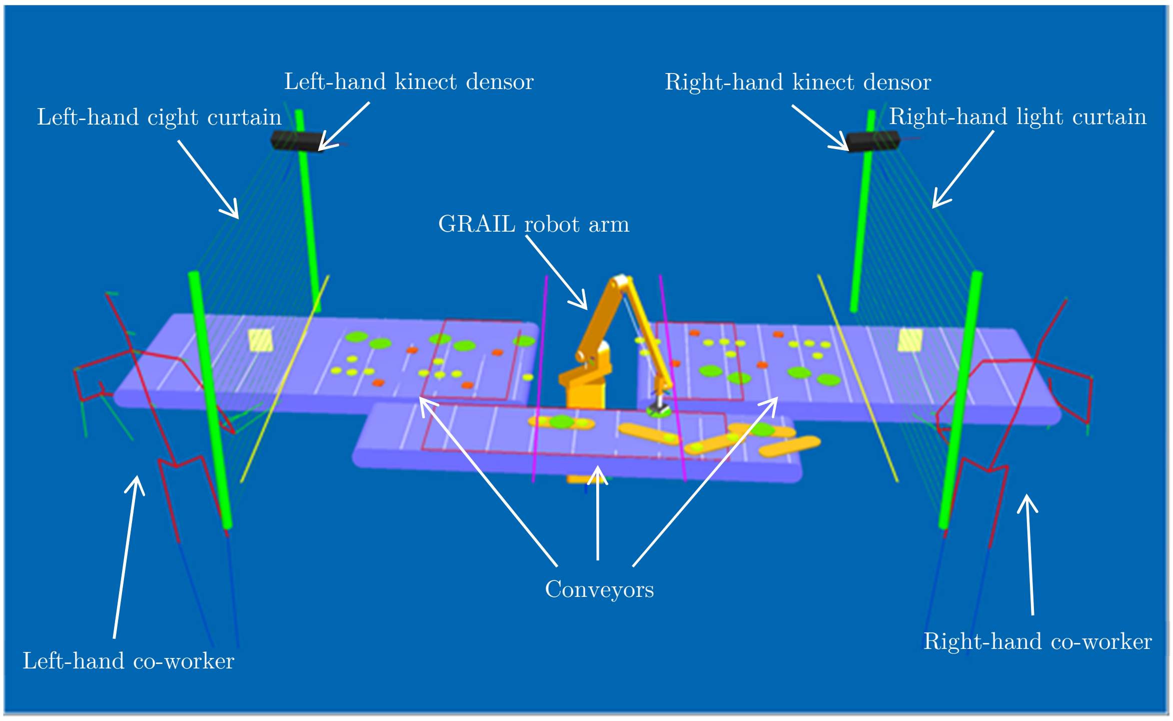


Figure 25: Annotated main simulator display.

The graphical simulation can also be displayed during runs using the physical system. In this case it mimics the behaviour of the various devices within the system and can provide additional debugging information. It is also possible to run the system in mixed modes of operation in which some devices are fully simulated, whereas other devices are based on real-world physical implementations.

The following sections describe each of the component parts of the GRAIL System that are modelled in the simulation.

4.1.1 GRAIL robot

The GRAIL robot has been modelled using an idealized realization of its forward and inverse kinematics (see [1], Section 5). This model does not take into account various real-world aspects including:

- Errors in construction of the robot which result in differences between its actual kinematics and the modelled kinematics.
- Imperfections in the joint control system which results in both static (due to sensor position errors, flexing of the structure due to gravity, and backlash) and dynamic errors in the joint positions (due to imperfections in the feedback

position control system causing velocity, acceleration and gravity dependent following errors and flexing when the joints are moving).

- Slight imperfections in the calibration of the robot frame of reference relative to the other components in the system such conveyors, light curtains and Kinect sensors.

However, it has been shown to be useful in practice for both debugging the software and estimating the performance of the real-world system and any departures from reality have not been significant.

4.1.2 Kinect sensors

Two Microsoft Kinect V2.0 sensors are used to provide input to the plan selector described in detail in Section 4.4.1. Within the GRAIL System, the Kinect sensor simulation has two modes of operation: mimic mode and replay from file mode.

When in mimic mode, the sensor data represented in the graphical simulation is a direct representation of the data acquired in real time from the physical sensor. Since the physical GRAIL testbed only has one Kinect sensor, this mode can only be applied to the left-hand Kinect sensor within the simulation.

When in replay from file mode, data is replayed from a file in a cyclic manner. First, the data must be captured to the file by using a special sub-mode that simply captures the raw Kinect data and saves it in a file. On replay, the captured data is treated in the same manner as if it came from the sensor in real time. When the data file ends, it is replayed from the beginning again so that continuous data is available.

Since two operators are needed in many test scenarios, the recorded data can be used twice: once to simulate the left-hand operator and once to simulate the right-hand operator. In the case of the right-hand operator simulation, the data is mirrored in a left-right manner so that the actions recorded by the original operator, who was working on the left, are appropriate for interacting with the ingredients and conveyors on the right. Additionally, a time offset is applied to the data used by the right-hand operator simulation so that the movements are not synchronised to the left-hand operator.

The Kinect sensor data can also be used to generate simulated light curtain activity and this is described further in Section 4.1.3. Visually, the Kinect sensor data is represented by “stick models” of the human co-workers as shown in Figure 26.

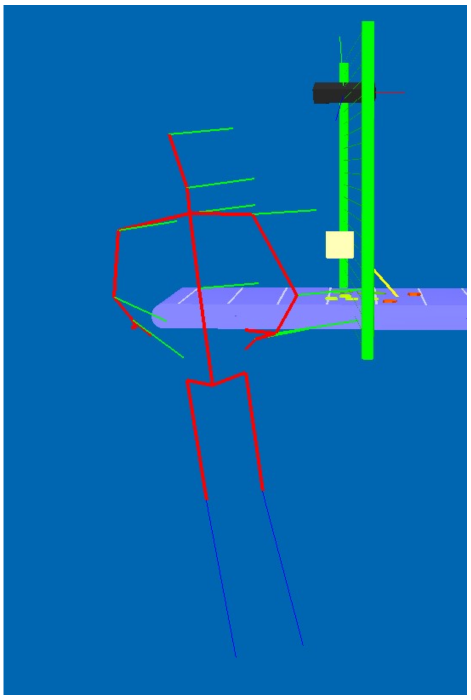


Figure 26: Stick Figure representing position of co-worker.

The colour of the "sticks" is used to indicate the reliability of the data originally recorded by the Kinect sensor. When the Kinect sensor can only see one end of a link between joints, the position of the unseen end is estimated. A link in which both end joint positions are seen is coloured red, and one in which only one end is visible and the other is inferred by the sensor is shown in dark blue. In general, the Kinect sensor cannot see the positions of the co-worker's lower legs and feet because the conveyors are in the way and so these are always inferred. This is not detrimental, however, since these body parts are not likely to enter the robot's workspace.

Additionally, the visual representation of the Kinect sensor data includes provision for depicting the predicted position of the operator at a future point in time. This predicted position is shown by the ends of the green velocity vector lines radiating from the stick figure's joints. The predictions are used in the plan selector (described in Section 4.4.1) to choose which ingredient to pick and where to place it. These predictions can be made in two different ways depending on the requirements of the tests being performed: using a Kalman filter to estimate velocity, or using "perfect" predictions, both of which are explained in the next two sub-sections.

4.1.2.1 Using a Kalman filter to estimate co-worker joint velocity and predict future co-worker positions

A standard linear Kalman filter is used to estimate the velocity of each joint based on new joint position measurements and its previous position and velocity at each measurement cycle. The following state vector was used for each joint:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

where $\mathbf{x}$ is the state vector, x, y and z are the components of the position of each joint, and $\dot{x}$, $\dot{y}$ and $\dot{z}$ are the components of the velocity for each joint.

The state evolves according to the following equation:

$$\mathbf{x}_{\boldsymbol{k}} = \begin{bmatrix} 1 & 0 & 0 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta_t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta_t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{\boldsymbol{k-1}}$$

where $\mathbf{x}_{\boldsymbol{k}}$ is the state at time k, $\mathbf{x}_{\boldsymbol{k-1}}$ is the state at time $k-1$ and Δ_t is the time interval between updates.

As can be seen from the state transition equation, The Kalman filter used for this task does not require any control inputs and so no control input model or input vector was necessary.

The following diagonal matrices were used for the measurement noise covariance and process noise covariance in the standard linear Kalman filter formulation:

$$\boldsymbol{R} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}$$

where $\boldsymbol{R}$ is the measurement noise covariance matrix and $\boldsymbol{Q}$ is the process noise covariance matrix.

The values for the measurement and process noise variance were chosen empirically to get the desired filter performance. Although not optimal, this proved to be good enough in practice as can be seen from Figure 27 which shows the measured joint position, the Kalman-filter-estimated position and Kalman-filter-estimated velocity when applied to a set of real data captured from a Kinect sensor. The grey line shows the velocity calculated by a simple differencing of position samples. It can be seen that the small amount of noise present in the original position data (blue, mostly hidden by the red line) has caused the calculated velocity to be very noisy and so the velocity calculated using this simple method would be unusable. The estimated position determined by the Kalman filter (red) is a good match to the actual position but with most of the measurement noise removed. In fact, it is such a good match to the original position data that the blue line is almost completely covered up by the red line in the graph. The green line shows the velocity estimated by the Kalman filter and is smooth enough to be used to extrapolate the future position of the operators and yet still follows the original noisy velocity curve closely in general shape, but without the noise excursions. It is important to note that the velocity curve produced by the Kalman filter is not significantly delayed compared to the original data.

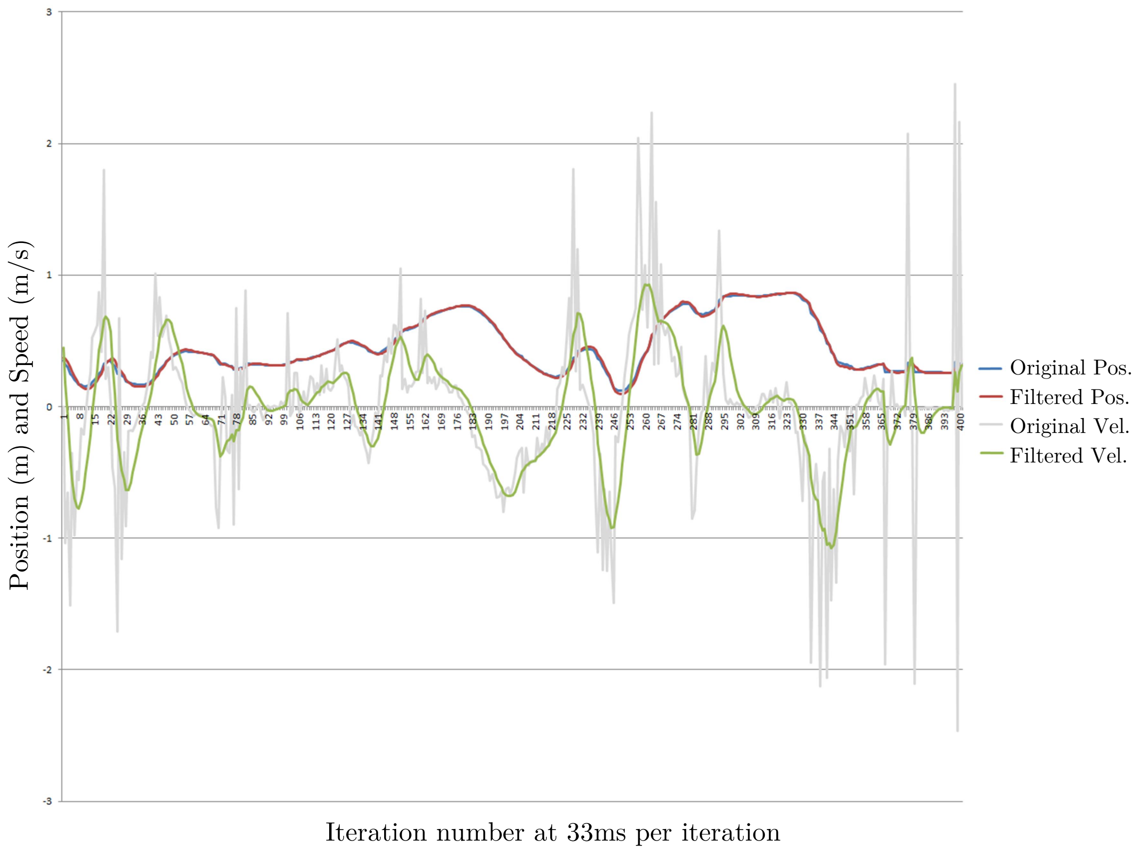


Figure 27: Kalman filter performance (position and velocity vs time).

4.1.2.2 Using “perfect” predictions

Since many of the experiments are performed using simulated input data for the Kinect sensors it is possible to calculate a “perfect” prediction of where the operator will be a short time in the future by simply reading ahead in the data file. This allows the performance of the system to be measured in idealised circumstances to ascertain the best case performance of any plan selector algorithm.

To simulate a “perfect” prediction the software merely has to look ahead in the stored Kinect data to determine where the operator’s joints will be at the desired future time.

4.1.3 Light curtains

The light curtains have been simulated in conjunction with the Kinect sensor simulation. The left-hand light curtain simulation can either mimic the state of the physical left-hand light curtain or it can work in full simulation mode. In full simulation mode, when the position of the operator, as measured by the Kinect sensor (either using real data or replayed data) is determined to pass through the plane defining the light curtain, the simulated light curtain is activated.

Visually, the light curtains are shown in the graphical simulation as an array of green dashed lines which change to red whenever the light curtain light beams are interrupted when using data from the physical sensor, or they are activated by the Kinect sensor when operating in simulation mode (see Figure 28).

Also, a small flesh-coloured rectangle is used to represent the potential position of a human hand. When the light curtain is interrupted, the “hand” will start to advance forward at a predetermined rate. By default the rate has been set at 1.6m/s which mimics the fastest movement speed that is reasonable according to the EN ISO 13855:2010 [4]. This is discussed in more detail in Section 4.3.1.

4.1.4 Conveyors

Three moving conveyors are shown in the simulation and can be made to either reflect the movement of the physical conveyors using the data from their optical position encoders, or can be simulated to move at a fixed speed.

When appropriate, the conveyor simulation can affect the position of any ingredients or bases placed on it so that they are seen to move realistically.

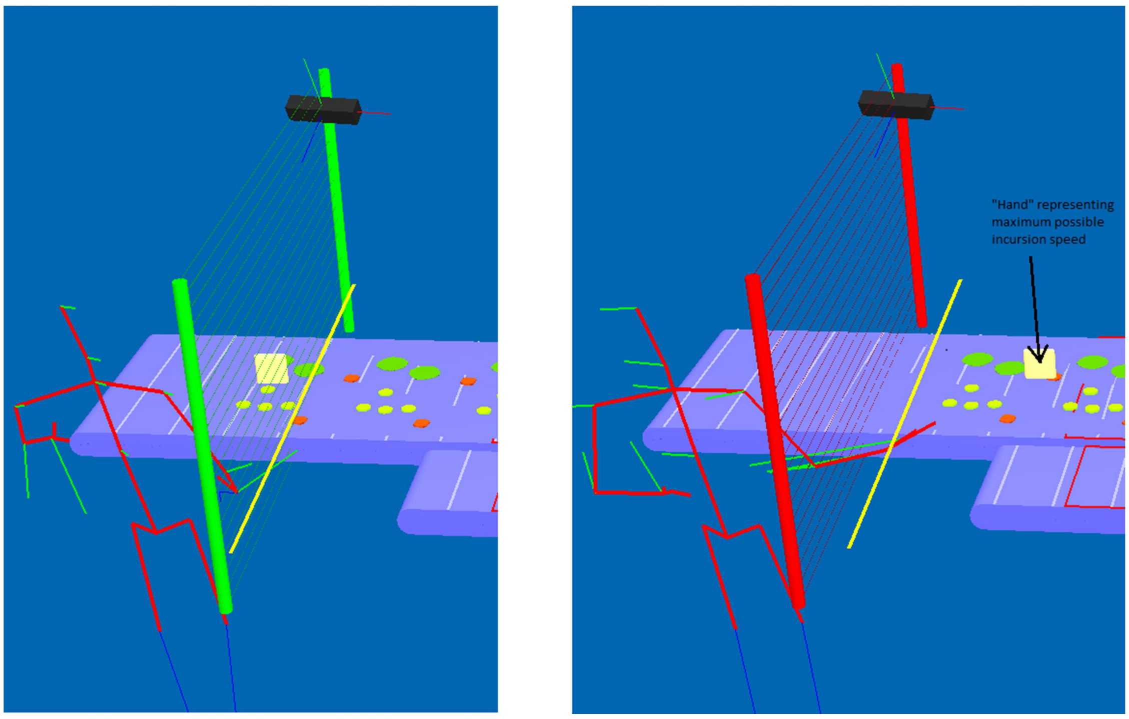


Figure 28: Un-activated (left) and activated (right) light curtain simulation.

4.1.5 Ingredients and bases

A simulation of the vision system generates new ingredients and bases and places them on the ingredients and bases conveyors, respectively. The ingredients and bases are depicted visually in the simulation as appropriately coloured shapes with the correct thickness. Since the current physical implementation of the system does not contain a real vision system, the simulated vision system is the only means of generating ingredients and bases which are input to the system even when the GRAIL testbed (described in Section 4.2) is used. When physical ingredients are used, their real-world positions must be made to match the positions generated by the simulated vision system for consistency.

As mentioned in Section 4.1.4, the simulated ingredients follow the positions of the simulated conveyors so that they can be seen to move correctly. When ingredients reach the ends of the two conveyors they are deleted and vanish from the graphical simulation.

4.1.6 Gripper

The robot gripper and its capabilities are simulated as well. In order to make a useful tool for software development purposes, the action of the gripper when picking ingredients has been made to be as independent as possible from the calculations within the rest of the software system. When activated, the gripper simulation software will search around the gripper within a radius of 30mm to find an ingredient that is present in the simulation. If one is found, the ingredient is "picked up" and becomes attached to, and controlled by, the position of the gripper. This means that when the robot moves, the ingredient is seen to move with the gripper.

When the gripper releases an ingredient, the ingredient "falls" to any conveyor that is underneath it when it is released. It then follows the movements of the conveyor. If an ingredient is released when it is not above a conveyor, it just becomes fixed in space.

4.2 The GRAIL testbed

The GRAIL testbed is a physical implementation of the GRAIL System and is used to test, debug and evaluate the performance of a real GRAIL System.

Figure 29 shows the GRAIL testbed and it can be seen that it contains the light curtain and Kinect senor for the left-hand co-worker only. During test runs, the right-hand co-worker and associated sensors are replaced by simulation versions as described in Section 4.1.2 and Section 4.1.3.

The left-hand ingredients conveyor and the bases conveyor are replicated using physical conveyors which transport the ingredients and bases on a moving platform rather than a continuous belt. The right-hand ingredients conveyor (not seen in the photograph) is implemented using a stationary platform and so during tests using the testbed, the right-hand conveyor speed must be set to 0 if physical ingredients are to be used so that the motion of the virtual ingredients in the simulation matches the motion of the ingredients in the real world.

The right-hand Kinect sensor and light curtain are also not present in the physical system, but can be simulated when required.

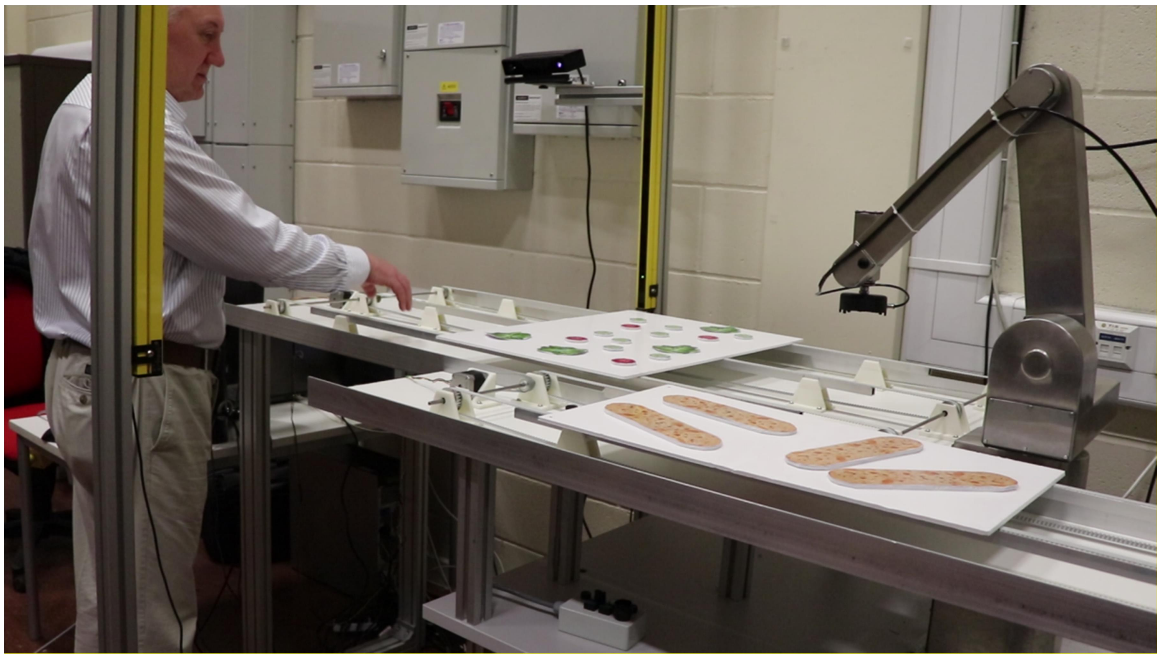


Figure 29: The GRAIL testbed.

4.3 The application of the UnCoVerCPS online verification approach

To overcome the limitations of traditional robot safety systems, the online verification safety approach that has resulted from the UnCoVerCPS developments has been applied to the existing GRAIL Robot Control System. In particular, the method of pre-verifying short segments of the robot's motion to ensure that they cannot intersect with the possible movements of any nearby co-workers has been adopted. The following sections describe the main software modules that have been developed during the UnCoVerCPS project and how they are incorporated into the original GRAIL Control System.

4.3.1 The verifier

The verifier software module is the embodiment of the UnCoVerCPS online verification approach in the GRAIL Robot Control System. The primary purpose of the verifier is to verify that proposed movements of the robot are safe given the data gathered from sensors about the actions of any nearby co-workers. To do this, the motion of the robot is divided up into short, 32ms long time segments (which are called "motion segments" in the remainder of this document) which are passed to the verifier for verifying prior to being acted on by the robot. At the same time as the new proposed motion segments are being verified, previously successfully verified motion segments are being acted on by the robot.

Additionally, the verifier has a secondary task; it must generate the "safe stop" trajectories which are appended to all motion segments prior to verification. The safe stop trajectories must bring the robot to a halt without exceeding any of the physical constraints of the robot. These physical constraints include position limits, velocity limits, acceleration limits, and jerk limits.

The new composite motion segments containing the task-directed part of the motion followed by the safe stop trajectory are then verified to ensure that they are guaranteed to be safe based on the input data from external sensors.

Due to the requirement to develop a low-cost system, the input to the verifier takes the form of data from two light curtains, one on either side of the robot. Light curtains have been chosen because they are relatively inexpensive, even when the high integrity requirements of this safety application are met. However, these are not used in a standard manner (whereby triggering would cause an E-Stop circuit activation). Rather, the output of the light curtain is taken as a digital signal input to the control system. The verifier uses this simple binary data from the light curtains to determine how far the human co-workers could have intruded into the workspace of the robot at each instant in time. It does this by using the maximum realistic speed of movement of the co-workers. A figure of 1.6m/s has been adopted as defined in EN ISO 13855:2010 [4]. During the time that a light curtain is activated, the human occupancy region is considered to expand towards the robot at this speed. The motion segments are checked against this region of space that could be potentially occupied by the co-workers. If there is no overlap, the motion segment is deemed to be safe, and if there is an overlap it is unsafe. Note that this is not based on any kind of *prediction* concerning the movements of the co-workers, but is based on their realistic worst-case behaviour and so motion segments that are deemed to be safe by the verifier can be executed without any further checking.

Of course, this method is very pessimistic since it assumes that co-workers always move at their maxim realistic speed. This could result in many needless stoppages of the robot. To mitigate this, a number of alternative motion segments are sent to the verifier on each verification cycle allowing the verifier to find a safe motion segment from amongst the set of all the motion segments that are sent to it.

These alternative motion segments arise because, at any instant in time, there are many ingredients within reach of the robot and, for each ingredient, there are a number of alternative locations on the bases where they could be placed. Throughout this document the picking of a particular ingredient and its placement in a particular location on a base is called a “plan”. Note that not all possible combinations of ingredients and place locations are a valid plan since some ingredients can only be placed in certain positions and some positions can only be filled when other ingredients in lower “layers” have been placed. See Figure 30 which gives an example of a set of valid plans.

At the start of each pick-place cycle the valid plans that could be chosen for execution by the robot are ranked in priority according to their effectiveness at achieving the goal of completing sandwiches. Then, the first motion segments for each of these ranked plans are sent to the verifier for verifying. The motion segment from the highest ranking plan that is deemed to be safe by the verifier is then queued for execution by robot subsystem.

On subsequent steps, the next motion segment for all plans that remain valid are verified. Note that some of the alternative plans may cease to be valid during plan execution since, for example, once a particular ingredient is picked, alternative plans involving different ingredients are no longer valid.

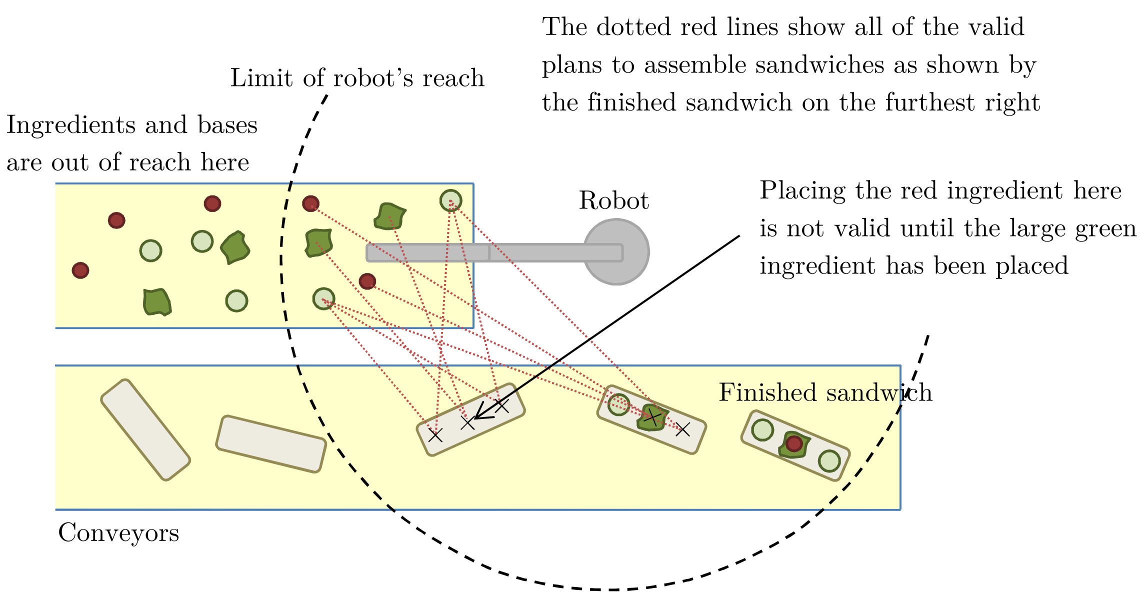

The dotted red lines show all of the valid plans to assemble sandwiches as shown by the finished sandwich on the furthest right
Limit of robot's reach
Ingredients and bases are out of reach here
Robot
Placing the red ingredient here is not valid until the large green ingredient has been placed
Finished sandwich
Conveyors

Figure 30: An example of a set of all valid plans.

As an example, consider the following scenario. At the beginning of a pick and place cycle, a number of valid plans are identified and sent to the verifier. The highest ranking safe plan is then acted on by the robot and so it starts to move towards an ingredient that it is going to be picked up.

On the next verification cycle, the next motion segments for this plan is sent to the verifier along with a set of alternative motion segments that would cause the robot to move away from its current plan and direct its movement towards an alternative pick location. The original plan is always given the highest priority at all times and so will be followed to the end if it remains safe. However if the verifier calculates that the motion segment for the original plan is no longer safe, the motion segment from the highest ranked alternative plan, which might be to pick a different ingredient or place it in a different location, will be verified and, if safe, will be executed instead.

A flow chart showing how the verifier is integrated into the GRAIL Robot Control System is shown in Figure 31. The pipelined manner in which the system operates can be clearly seen. While the verifier is performing its functions, the previously verified motion segment is being executed by the robot arm. The architecture takes advantage of the 8 physical cores within an Intel Xeon E5-2650 octa-core 2.60GHz processor to verify up to 8 alternative motion segments at the same time, thereby allowing the verification process to occur in real time.

It can also be seen that if a segment ends, but a new, verified safe motion segment is not available, the robot motion continues by executing the safe stop motion appended to the end of the last motion segment that was executed.

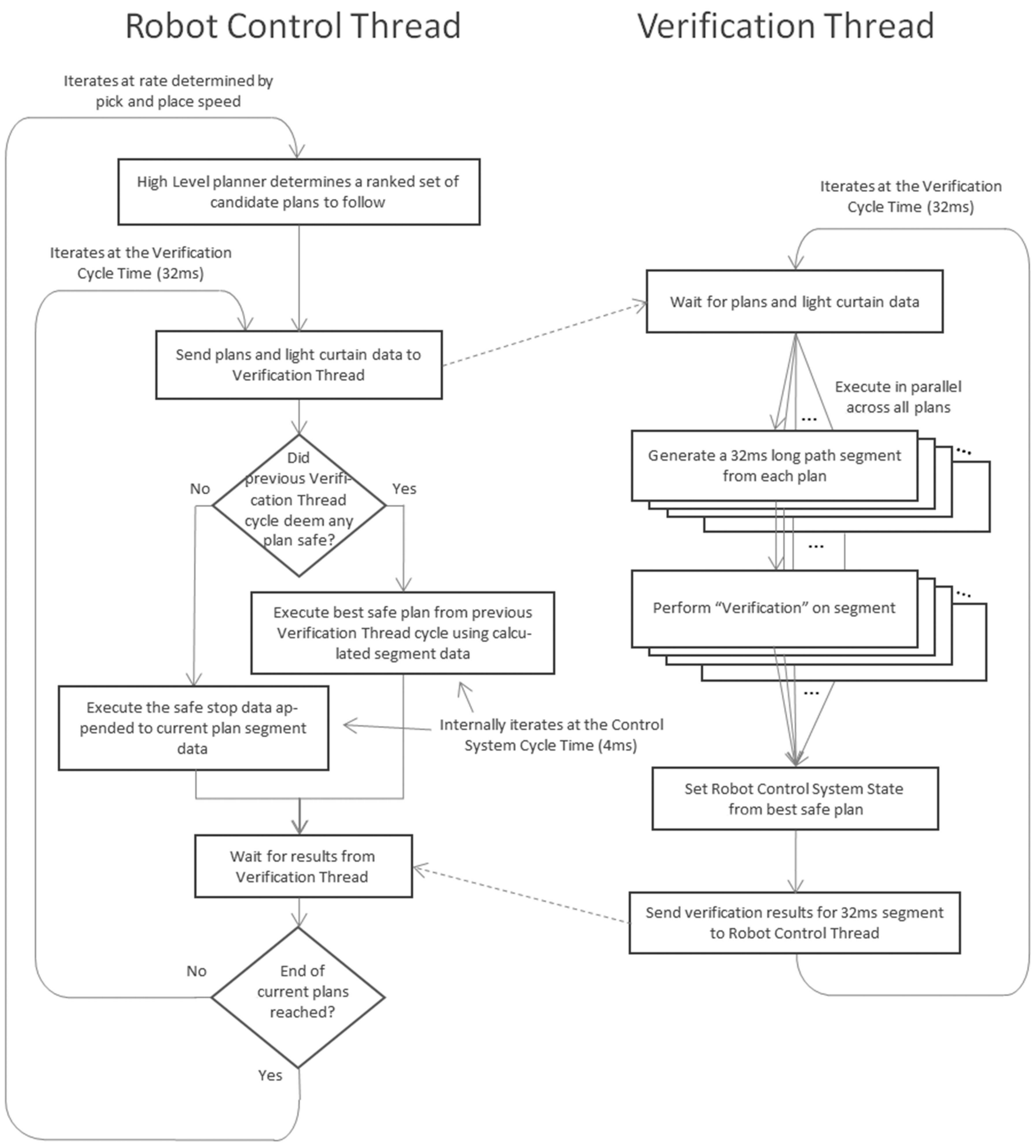

Robot Control Thread
Verification Thread
Iterates at rate determined by pick and place speed
High Level planner determines a ranked set of candidate plans to follow
Iterates at the Verification Cycle Time (32ms)
Send plans and light curtain data to Verification Thread
Did previous Verification Thread cycle deem any plan safe?
No
Yes
Execute best safe plan from previous Verification Thread cycle using calculated segment data
Execute the safe stop data appended to current plan segment data
Internally iterates at the Control System Cycle Time (4ms)
Wait for results from Verification Thread
End of current plans reached?
No
Yes
Iterates at the Verification Cycle Time (32ms)
Wait for plans and light curtain data
Execute in parallel across all plans
Generate a 32ms long path segment from each plan
Perform "Verification" on segment
Set Robot Control System State from best safe plan
Send verification results for 32ms segment to Robot Control Thread

Figure 31: Flowchart of GRAIL Control System architecture incorporating the verifier only.

4.3.2 Testing the verifier in simulation

This section discusses the testing that was performed using the simulation of the new GRAIL architecture using the verifier module alone. The goal of the tests described in the remainder of this section is to test the functionality just described and to determine the change to the efficiency of the GRAIL System that results from using the verifier to maintain safety when the system is operated in the presence of nearby co-workers.

The verifier provides two main features which must be tested. First, it is responsible for stopping the robot when a human co-worker gets too close in order to maintain safety. However, to provide maximum system throughput, stopping should be minimised within the conservative nature of the safety assumptions. Second, when then verifier is used without support from additional software modules which choose a suitable plan (see Section 4.4), it has the function of finding a safe plan amongst the alternative plans sent to it by the high-level controller as described in Section 4.3.1.

The following sections describe how each of these features was tested using the simulation.

4.3.2.1 Tests showing the verifier choosing between plans in simulation

Several simulation runs were performed in which there where a surplus of ingredients to pick and positions in which they could be placed. This ensured that the verifier was presented with many alternative plans on each pick and place cycle. The simulation of the left co-worker was configured so that interruptions to the light curtain would occur frequently. Information generated as the simulation was running was logged to determine if an alternative plan was chosen when the primary plan was deemed to be unsafe by the verifier.

It was found that there were no instances recorded in which the primary plan was deemed to be unsafe, but an alternative plan was safe. Figure 32 shows the position of the robot's joint 1 when it is executing a typical motion segment. The black vertical line shows the point at which an alternative plan must be chosen because the new motion segment from the original plan is no longer safe. In this example, the alternative plan has a target position which is very distant from the original target position. It can be seen that it takes a finite amount of time for the new path to take effect since the motion of the robot has to honour dynamic constraints such as maximum acceleration and jerk. In fact, the magnified portion of the graph shows that over the time duration of the next motion segment, there is only a miniscule change between the original and alternative paths. This means that, as far as the verifier is concerned, if the original motion segment was unsafe, then the alternative motion segment is almost certainly unsafe as well.

When the fact that the example shown had a primary and alternative motion segment from plans that had widely different ultimate target positions is taken into account, it can be seen that, in general, the verifier is unable to choose between plans based on their safety over the short duration of a motion segment because, when the motion segment from the primary plan becomes unsafe, the motion segments from all the alternative plans will be unsafe as well. This somewhat negative result is discussed further in Section 4.4 below.

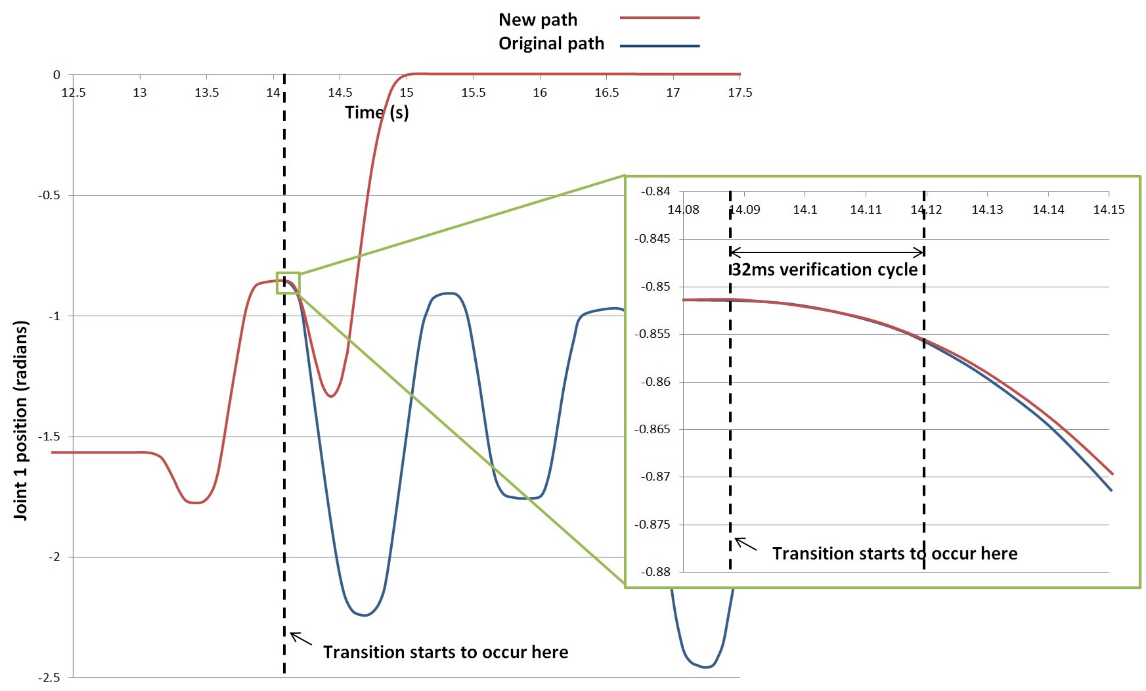


Figure 32: Difference between the primary motion segment and an alternative motion segments.

4.3.2.2 Tests showing the verifier providing safety activity in simulation

For this test, simulations were run in which the simulated conveyors had a surplus of ingredients and bases and so the robot remained active. The simulated left-hand co-worker frequently interrupted the light curtain. Sometimes the interruption was for a short period and sometimes for a longer period. The verifier was given only a single plan for each pick and place cycle and so whenever the motion segment was deemed to be unsafe, the movement of the robot had to stop.

Each time the light curtain was interrupted, the proximity between the robot and a hypothetical "hand" moving at the maximum realistic speed of 1.6m/s as specified in EN ISO 13855:2010 [4] was logged. When the robot did not stop during the light curtain activation, the minimum clearance between the robot and the "hand" during the entire time that the light curtain was activated, was recorded. If the robot stopped, the minimum clearance during the time between the light curtain being activated and the robot stopping was recorded. Additionally, when the robot stopped, the minimum clearance for the entire light curtain activation duration was also recorded.

The results from this test are shown in Figure 33. The horizontal axis is simply the index of the particular light curtain activation instance and so the results of each of the 667 light curtain activations during the test run are shown from left to right on the graph. Note that the horizontal axis is not equivalent to time since there were arbitrarily long time durations between light curtain activations.

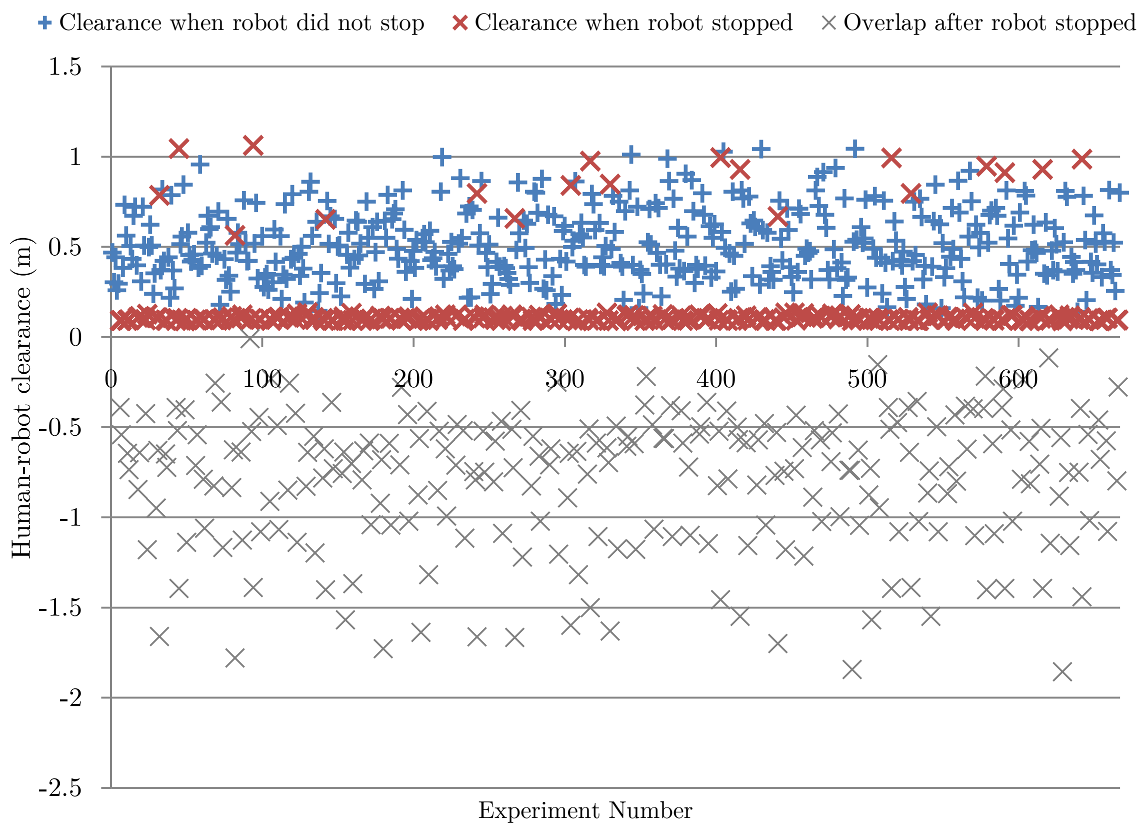


Figure 33: Robot-to-hand clearance during light curtain activation.

For each instance of a light curtain activation, one or two points are plotted against the vertical axis. When the robot did not stop, a single point is plotted (a green +) to show the minimum clearance achieved. Since the verifier did not deem the motion to be unsafe, all of these points should lie in the positive half of the graph since there should have been a clearance between the robot and the worst-case "hand" at all times.

When the verifier caused the robot to stop, two points are plotted on the graph. The first point (a red **x**) shows the minimum clearance between the robot and the hand during the time that the robot was moving. Again, these should always lie in the positive half of the graph since there should be clearance between the robot and the "hand" during the time that the robot is moving.

It can be seen that most of these points lie close to a horizontal line just above 0.0m. This is because the verifier has caused the robot to stop only when absolutely necessary because a potential collision was impending. The gap between the points and the *clearance=0.0* line is due to the assumed latency of the real-world light curtain activation (55ms). In the simulation, the actual light curtain latency is 0ms and so the allowance for this assumed latency, combined with a "hand" speed of 1.6m/s results in a gap of 0.088m.

It can be seen that a few of these point do not lie close to 0 but have moderate positive values. This occurs when the light curtain activation stopped (i.e. the "hand" retreated behind the light curtain) in the time between the verifier performing its calculations on the motion segment in advance of it being executed by the robot, and the robot actually executing it.

The second point (a grey **x**) shows the clearance during the total duration of the light curtain activation, including time after the robot stopped and so was safe. Since the verifier deemed the motion segment to be unsafe, causing the robot to stop, these should always lie in the negative half of the graph in order to have triggered the need to stop.

Since all the points on the graph lie in the required regions, this demonstrates that the verifier was correctly performing its safety function.

4.3.2.3 Tests validating the safe stop trajectory in simulation

When the verifier analyses a motion segment it appends a "safe stop" trajectory to it in case it is necessary for the robot to stop when the motion segment ends. The profile of the safe stop must meet a number of criteria since it must be within the capabilities of the robot to execute it correctly and it must be compatible with holding ingredients in the gripper.

The graphs shown in Figure 34 show an example of the radial, tangential and vertical position, velocity, acceleration, and jerk against time when a safe stop is executed.

It can be seen in this example that the robot stops in a controlled manner without exceeding the velocity, acceleration or jerk limits. Although this is only a single example, the simulator constantly monitors, positions limits, velocity limits and acceleration limits during execution and would cause a fatal error to be raised if any were exceeded, so, it can confidently be stated that a correctly controlled safe stop has been executed many times without error.

4.3.2.4 Tests to measure the efficiency of the system using the verifier alone in simulation

To measure the efficiency of the system using the verifier alone, a long duration simulation run (1000 seconds) was performed during which a number of statistics were gathered. During the first run, there were no nearby co-workers and so the robot could continue to move without any stoppages. This represents the best possible efficiency of the system.

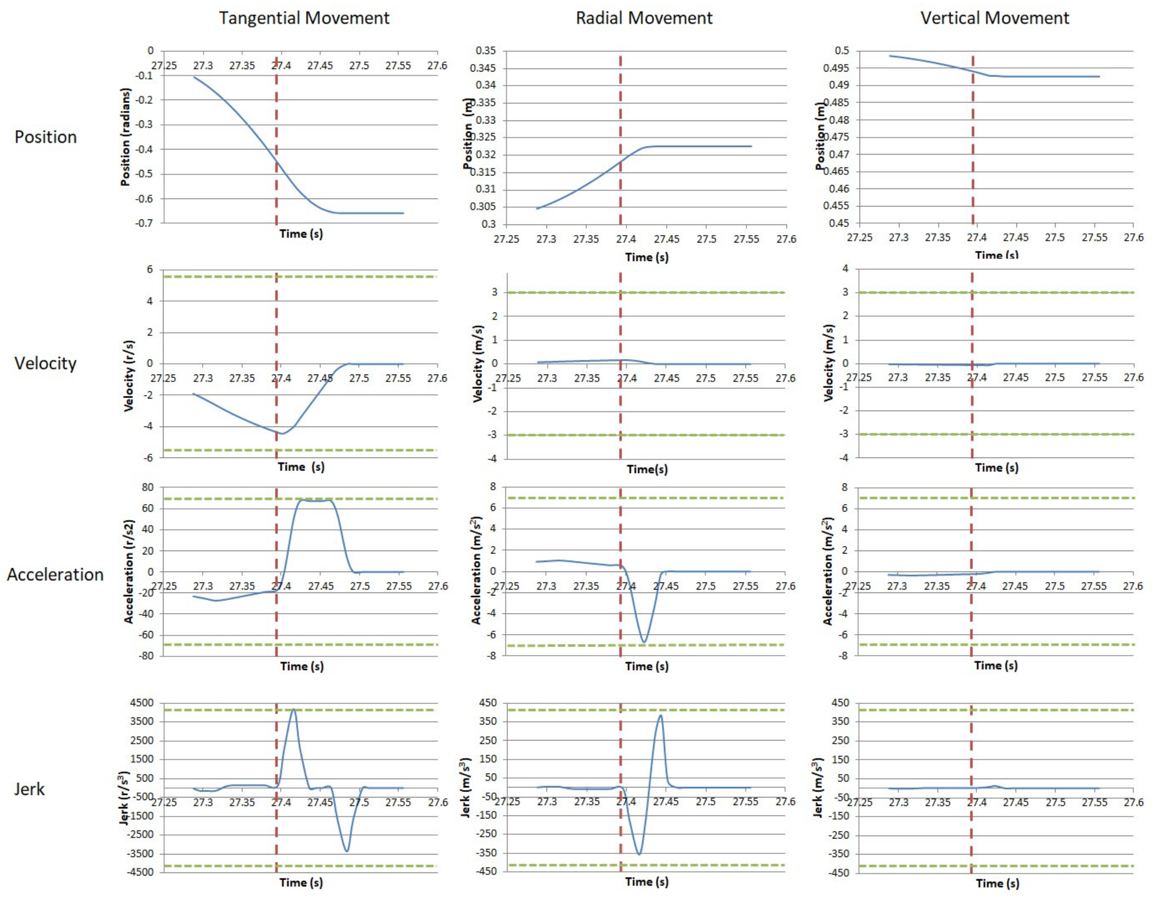


Figure 34: Example movement profile during a safe stop with the red line showing where the safe stop starts and the limits are marked in green – simulation.

In the second test run, simulated co-workers on either side of the robot each break the light curtain at a rate of approximately once every 6 seconds. In this test run, the verifier was used to maintain safety by stopping the robot when required to prevent collisions.

In the third test run, simulated co-workers were again present. However, in this case, the verifier was not used to stop the robot intelligently, but instead, the robot was stopped whenever the light curtains were activated. The intention of this test was to gather statistics from a system that operates as if a simple E-Stop occurred when the light curtain was breached. However, to make this run more comparable with the data obtained when the verifier was used, a very optimistic stoppage time of only 1 second was imposed every time the robot stopped even though, in practice, the stoppage time would be much longer than that when an E-Stop occurs since the robot would have to be restarted and, in many cases, rehomed. **Table 1** shows the results obtained from each of these runs.

Table 1: Performance figures with/without nearby co-workers – simulation.

	Placed ingredients	Number of completed sandwiches	Number of rejected ingredients	Number of interruptions	Number of light curtain breaches
Without nearby co-workers	710	155	0	0	0
With nearby co-workers and verifier	452	89	23	81	344
With nearby co-workers and simple E-Stop	119	6	86	167	353

It can be seen that the efficiency of the system with nearby co-workers is reduced to 64% (based on number of placed ingredients) when compared to running the system without nearby co-workers. However, this is significantly better than the efficiency when a simple E-Stop arrangement is used, which has an efficiency of much less than 17%.

Using the verifier to intelligently choose when to stop the robot and when the robot can continue moving is, therefore, shown to provide much higher efficiency than using a simple E-Stop arrangement. However, the efficiency of the system using the verifier alone is still much reduced compared to a scenario in which human co-workers are excluded from the workspace of the robot by physical means such as barriers. In practice, of course, the light curtain activation rate by the co-workers would be far lower than was used to gather these results, since they would be trained to stay away from the robot when it was moving. However, the indicated trend would still remain true.

4.3.3 Real-world testing of the verifier

Similar tests to those described in Section 4.3.2 were repeated using the GRAIL testbed. The purpose of these tests was to demonstrate that the performance obtained was similar to the results obtained from running the simulation.

The design of the conveyors used made it difficult to run many ingredients and bases through the system continuously and so simulated ingredients and bases were used. In this mode of operation the ingredients and bases are generated automatically by the software without any input from a vision system. Of course the ingredients and bases do not have a physical realisation, but their virtual representations are simulated to move at a rate corresponding to the real movement of the conveyors. All the activities of the robot arm were the same as if real ingredients and bases were used and so there was very little loss in experimental fidelity by making this simplification.

Additionally, the physical implementation of the system does not contain a right-hand ingredients conveyor, light curtain or Kinect sensor and so these were also simulated.

A real left-hand Kinect sensor and light curtain were used, and, during the experiments, efforts were made by the test co-worker to duplicate the behaviour of the simulated right-hand co-worker, as perceived by the sensors, so that the results were broadly comparable to the results obtained from simulation.

4.3.3.1 Tests showing the verifier providing safety activity on the testbed

A test was performed that was very similar to the test described in Section 4.3.2.2 with the exception that the GRAIL testbed was used as described above.

To perform these tests, a video camera was used to record the movements of the robot arm and test co-worker when the light curtain was breached. The video frames were then analysed on a frame-by-frame basis to determine if and when the robot stops and the distance between the robot and co-worker using the reference distance scale placed in the image. Figure 35 shows the general setup that was used to capture this data and illustrates the correction necessary to each measurement due to parallax effects.

Due to the practicalities of running this test, fewer data points were gathered than when the simulation was used. The values of “overlap after the robot stopped” were not recorded since these have no significance unless the human speed is maintained at exactly 1.6m/s which could not be achieved in practice.

The test co-worker was instructed to move his hand at speeds between a normal pick and place speed and a fast, but not abnormally fast speed. The results obtained can be seen in Figure 36. It can be seen that the clearances were always positive, i.e. there were never occasions when a collision between the robot and the co-worker’s hand could have occurred. Again, it can be seen that there are a band of clearances distances between 0mm and 100mm that are not seen to occur. This is due to the 55ms allowance (resulting in 88mm clearance at 1.6m/s) made for the worst-case light curtain activation latency. In practice, the light curtain activation speed was much faster than this on average, but, for safety, the worst case had to be catered for.

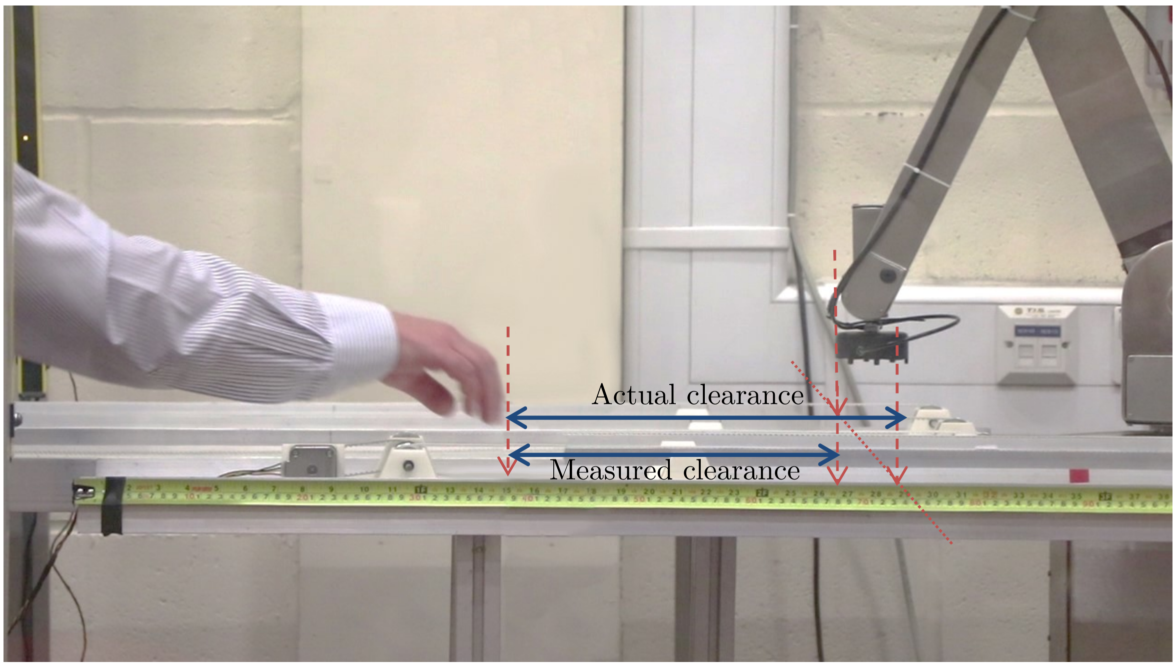


Figure 35: Experimental setup to measure verifier safety.

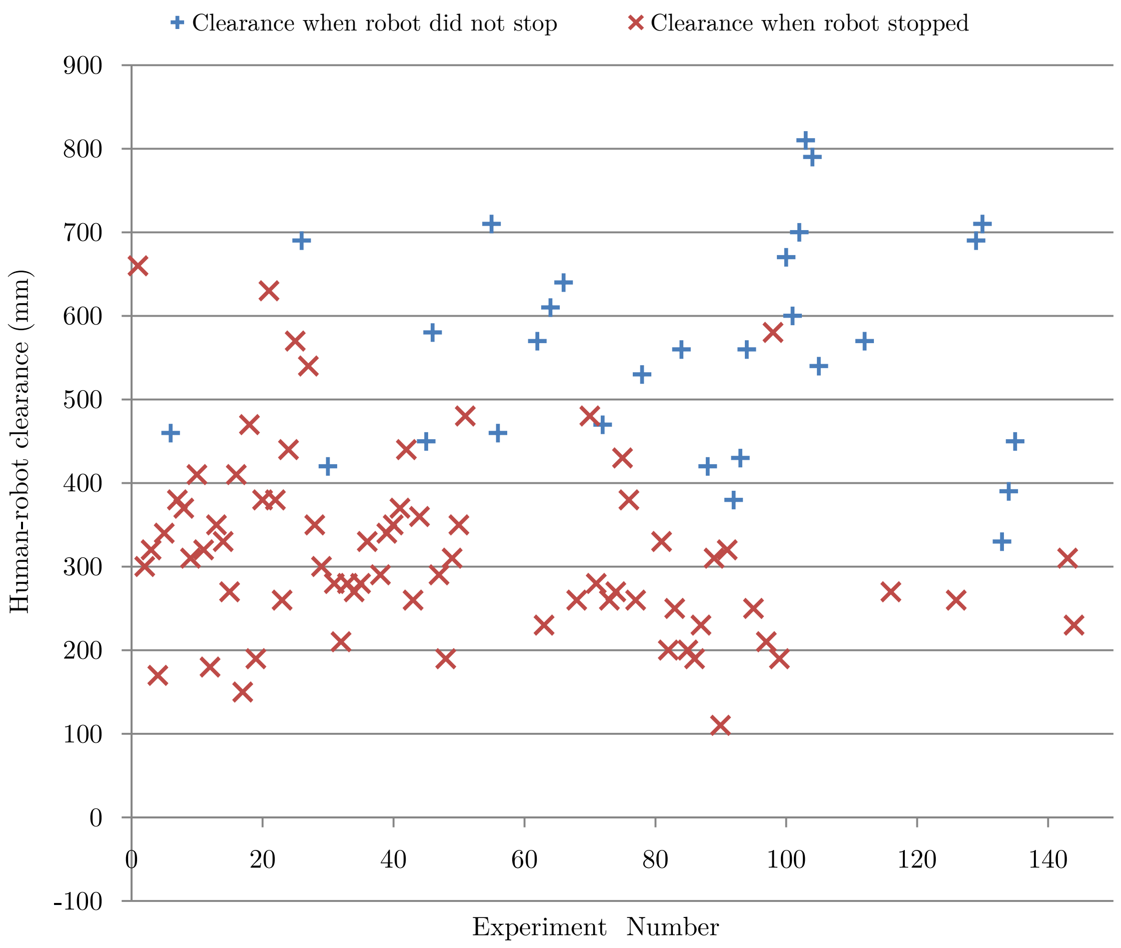


Figure 36: Robot-to-hand clearance during light curtain activation – testbed.

Further tests were performed in which the co-worker tried to move his hand towards the robot as fast as possible at speeds in excess of what would be considered a “normal” range of movement speeds. It was found that a few of these tests resulted in a negative clearance between the co-worker’s hand and the robot, i.e. a potential collision (note that actual collisions did not occur since, for safety, the co-worker’s hand was always in front of the robot’s workspace). For these tests, additional measurements were made from the video data to determine the speed of movement of the hand. Figure 37 shows the estimated hand movement speed against clearance that resulted from this test. It can be seen that all of the tests that resulted in a negative clearance, had a hand speed that was significantly greater than the modelled 1.6m/s maximum hand speed. In fact, only for hand speeds in excess of 3m/s were potential collisions (negative clearances) actually experienced. Again, this is due to the 55ms allowance made for the worst-case light curtain activation latency.

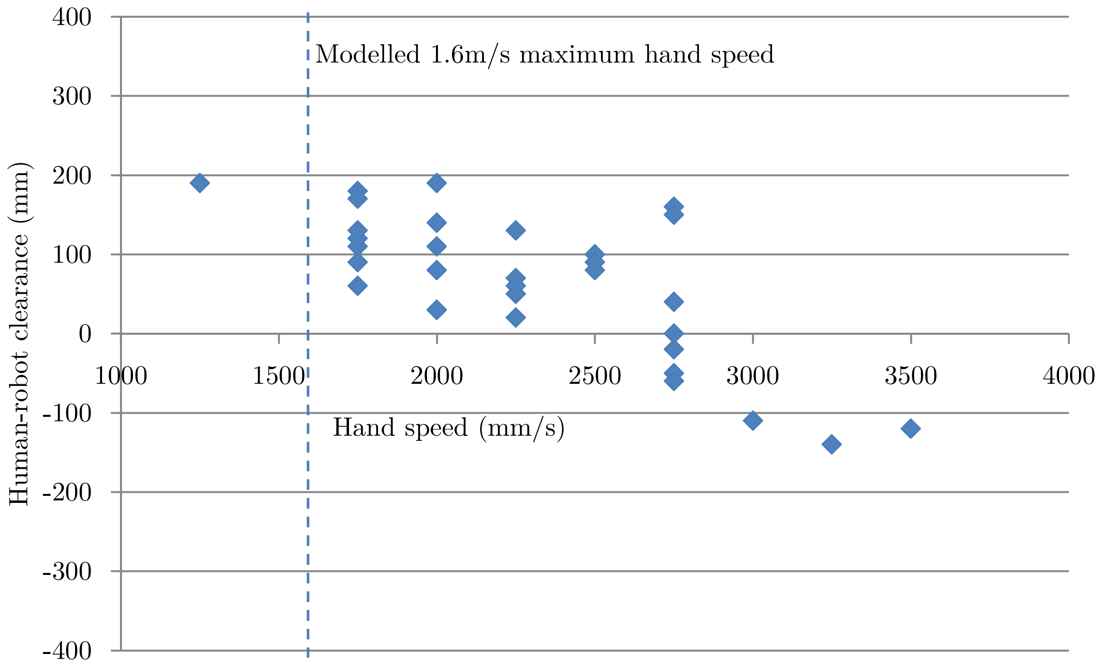


Figure 37: Human hand speed against human-robot clearance for high-speed moves – testbed.

4.3.3.2 Tests validating the safe stop trajectory on the testbed

Similar to the tests described in Section 4.3.2.3, results were captured from the testbed when a safe stop trajectory was executed. In essence this data is no different from the data gathered during the simulated test runs since the same software code is used to generate the data in both cases, but the test was repeated for completeness and the results can be seen in Figure 38. Again, it can be seen that all dynamic constraints were met during the safe stop.

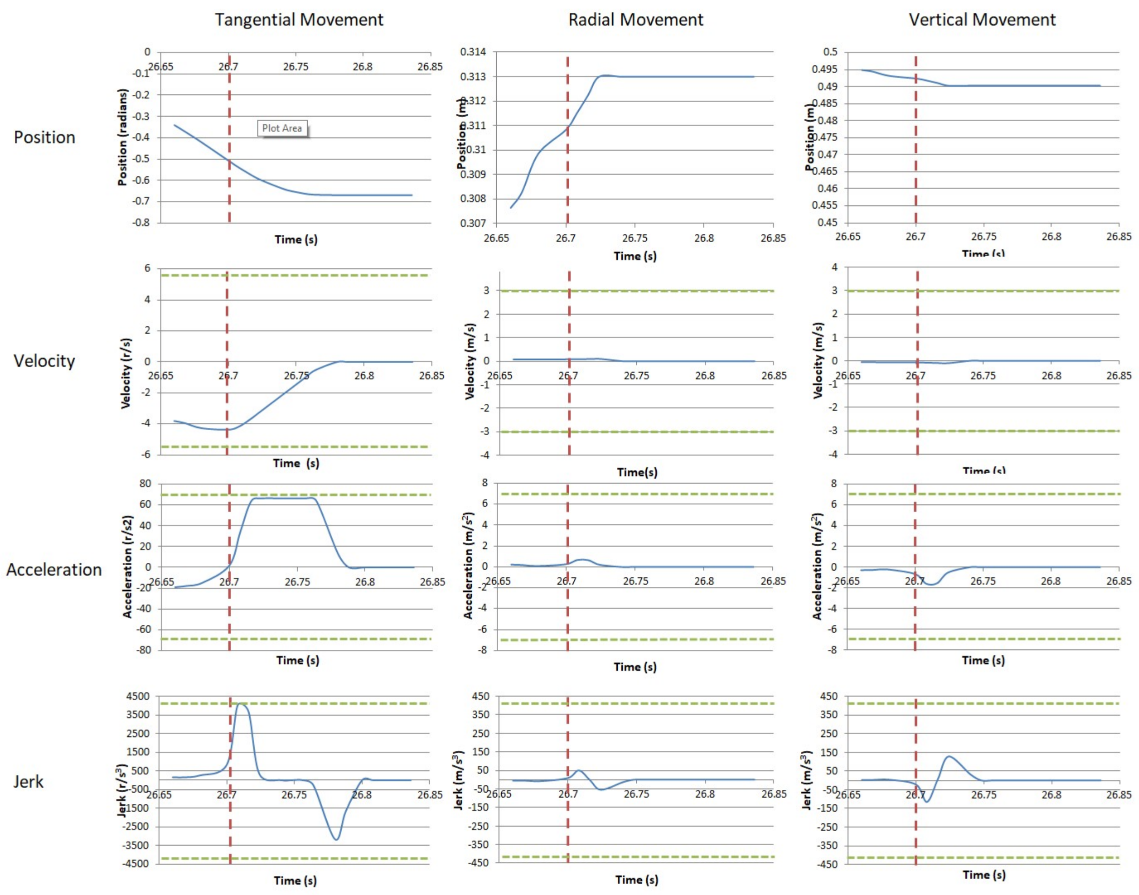


Figure 38: Example movement profile during a safe stop with the red line showing where the safe stop starts and the limits are marked in green – testbed.

4.3.3.3 Tests to measure the efficiency of the system using the verifier alone on the testbed

The efficiency of the system running on the testbed was measured in a similar manner to that described in Section 4.3.2.4 and results were obtained as shown in **Table 2**. A human was used for the left-hand co-worker, along with a real Kinect sensor and light curtain. In order to produce results that are somewhat comparable to the results obtained from the simulation, the test co-worker attempted to replicate the movements of the simulated right-hand co-worker as much as possible so the number of light curtain activations and Kinect sensor data was similar.

It can be seen that the results in the table are broadly similar to the results shown in **Table 1** that showed the results from the simulation. In real life, the performance reduction to 62% was seen due to the action of the verifier stopping the robot when there were nearby co-workers and an efficiency of 25% was measured for the comparable E-Stop case.

Table 2: Performance figures with/without nearby co-workers. The numbers in brackets show the results scaled to a 1000 second run – testbed.

	Placed ingredients	**Number of completed sandwiches**	**Number of rejected ingredients**	**Number of interruptions**	**Number of light curtain breaches**
Without nearby co-workers	169 (676)	38 (152)	0 (0)	0 (0)	0 (0)
With nearby co-workers and verifier	105 (420)	21 (84)	5 (20)	15 (60)	77 (308)
With nearby co-workers and simple E-Stop	43 (172)	5 (20)	14 (56)	41 (164)	75 (300)

4.3.4 Conclusions from testing the verifier alone

The results of the tests described above, conducted using both the graphical simulation and the GRAIL testbed, showed a good general agreement, even though there was difficulty in maintaining consistency and making measurements of the co-worker's movements.

The results showed that the verifier always stopped safely when it was required to do so, and didn't stop when collisions between the robot and co-workers were impossible even though the light curtain had been activated. The verifier therefore meets its primary purpose and is always safe as long as the co-worker's movement speed is limited to 1.6m/s. However the verifier, by design, operates in a very pessimistic manner and so its operation in the presence of human co-workers that overlap with the robot's workspace still resulted in more stoppages than are desirable.

Although it was hoped that the verifier would be able to choose appropriate plans to execute in the presence of potential collisions with co-workers, in practice this did not work very well since the various plans were insufficiently different over the short time duration of the motion segments.

To resolve this difficulty, a new software architecture was developed that used a "plan selector" software module to pre-select the plans that are most likely to be safe based on the available sensor data. Then, this one and only plan could be verified by the verifier in the usual manner, but with the expectation that it is less likely to be deemed unsafe. More details of the operation of the plan selector can be seen in Section 4.4.

4.4 Increasing the operational efficiency in the presence of human co-workers

As described in the previous section, using the verifier alone to choose a safe path resulted in an inefficient system that suffered from frequent robot stoppages when there were human co-workers in close proximity. The plan selector module was created to mitigate this issue.

4.4.1 The plan selector

As described at the end of Section 4.3.4, the plan selector is intended to increase the picking efficiency when the motion of nearby operators is predicted to intersect with the path of the robot. The scenario that is under consideration is one in which the GRAIL Robot has human operators working on either side of it, each of which could reach into the workspace of the robot. However, it is assumed that the left-hand operator can only reach the left-hand half of the robot's workspace and the right-hand operator can only reach the right-hand half of the robots workspace. This means, for example, that when the left-hand operator is predicted to intrude into the robot's workspace, the GRAIL Robot Control System can ensure that the robot works in the right-hand half of its workspace thereby preventing the need to stop. The mechanism works in a mirrored manner for the right-hand operator.

It is the responsibility of the plan selector to create this desired behaviour. It is able to do this because there are usually many options for suitable ingredients to pick and positions to place them as was shown in Figure 30 in Section 4.3.1 which described the verifier.

At the start of every pick, the plan selector considers all valid plans (ingredients to pick with their associated place locations) and determines the best one according to the results of an evaluation algorithm (the algorithm is described later in Section 4.4.2). Note that at the time of the pick, both the pick movement and the place movement are evaluated using the predicted behaviour of both the left- and right-hand co-workers. Once the pick has been completed, the place is re-evaluated since the behaviour of the co-workers could have changed, and the positions of the ingredients and bases will have changed slightly due to the movement of the conveyors.

Note that there are still occasions when the plan selector will not be able to prevent the robot from having to stop such as when there are predicted robot workspace incursions from both sides at the same time, or due to the uncertainties involved in predicting the behaviour of the human co-workers over the time duration of a pick and place operation.

The way in which the plan selector fits into the rest of the GRAIL Robot Control System is shown in Figure 39.

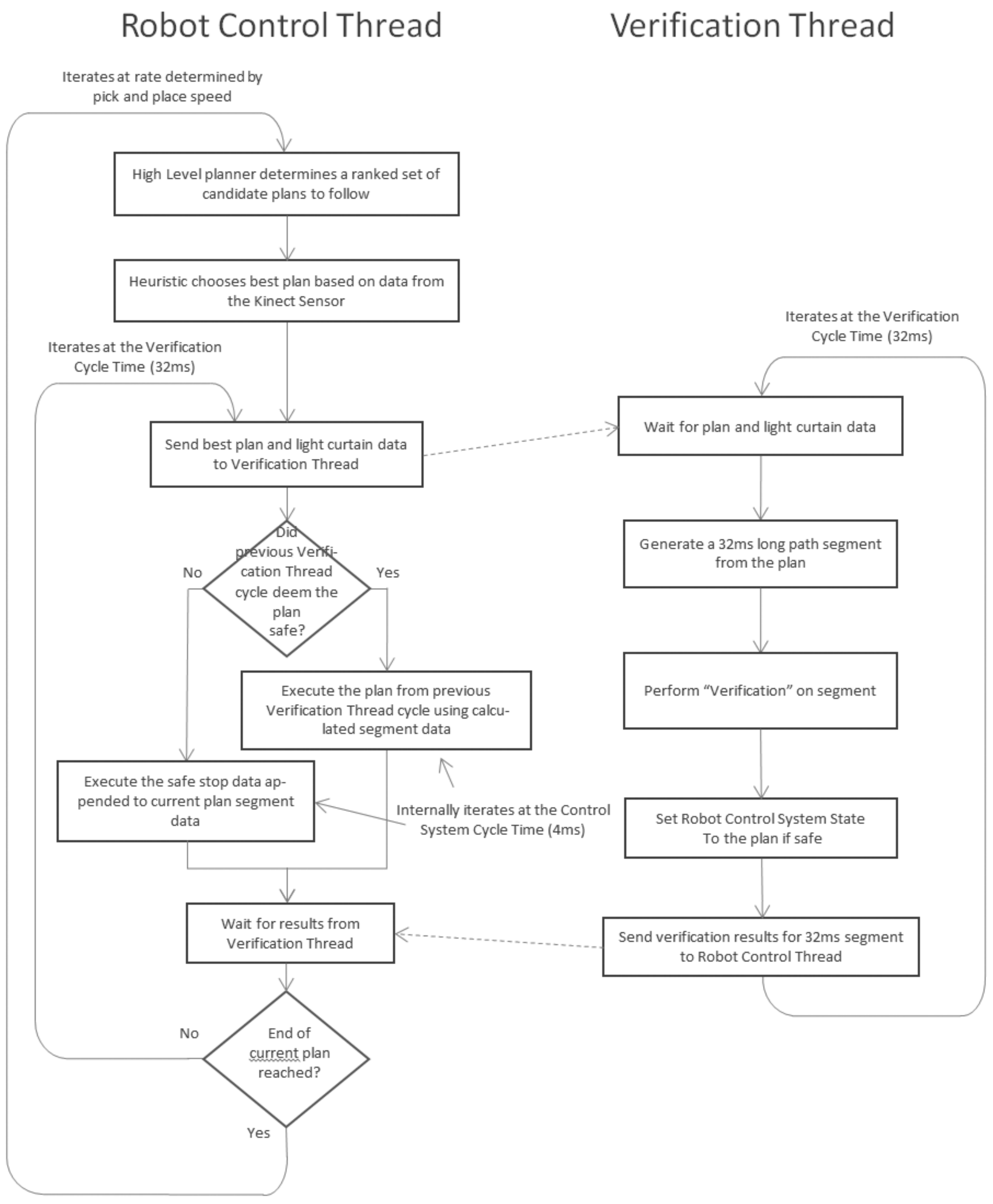

Robot Control Thread
Verification Thread
Iterates at rate determined by pick and place speed
High Level planner determines a ranked set of candidate plans to follow
Heuristic chooses best plan based on data from the Kinect Sensor
Iterates at the Verification Cycle Time (32ms)
Send best plan and light curtain data to Verification Thread
Did previous Verification Thread cycle deem the plan safe?
No
Yes
Execute the plan from previous Verification Thread cycle using calculated segment data
Execute the safe stop data appended to current plan segment data
Internally iterates at the Control System Cycle Time (4ms)
Wait for results from Verification Thread
End of current plan reached?
No
Yes
Iterates at the Verification Cycle Time (32ms)
Wait for plan and light curtain data
Generate a 32ms long path segment from the plan
Perform "Verification" on segment
Set Robot Control System State To the plan if safe
Send verification results for 32ms segment to Robot Control Thread

Figure 39: Flowchart of GRAIL Control System architecture incorporating verifier and plan selector.

An important point to note is that the verifier is still used to ensure safe behaviour when there is an impending collision between the robot and a co-worker. The plan selector is intended to choose plans so that collisions are avoided as often as possible in order to increase operational efficiency. It is not intended to increase safety. Safe operation is already ensured through the use of the verifier. An advantage of this is that the plan selector software and sensors do not form part of the safety system and do not need to meet the same high integrity requirements that are required for the verifier and its associated sensors.

4.4.2 Plan selector co-worker motion prediction

A key requirement for the successful operation of the plan selector is to be able to generate accurate predictions of the motion of the co-workers for the time it takes to perform a pick or a place operation (in the range of 1 – 2 seconds). Since the sensors do not need to have high integrity, a sensor type that returns high quality, detailed information on the movements of the co-workers can be used. In practice, Microsoft Kinect V2.0 3D sensors were used because they provide detailed position information about the co-worker's limbs, head and joints at a 30 fps rate. Although these are only consumer quality devices, many industrial equivalents are available and could be used instead to achieve the same results.

A function of the plan selector is to take the data returned by the Kinect sensor and use it to predict the position of the co-worker in 1 second's time. To do this it calculates the velocity of each of the co-workers joints (here, a joint refers to each of the nodes in the human model and so the top of the head is also a "joint", for example) and then extrapolating the position of the joints based on the calculated velocity vector. As described in Section 4.1.2 this has been done by two different methods: determining velocity using a Kalman filter and "perfect" prediction.

4.4.3 Plan selector evaluation function

The plan selector chooses a plan by passing all the plans through an evaluation function. The evaluation function takes a number of factors into account when choosing the best plan as follows:

- Human-robot proximity – the minimum distance between the predicted human positions and the robot positions when it is executing the plan in question. See Appendix A for a description of how the calculation is performed. A higher number is better.
- Path length – the distance that the robot has to move to carry out the plan. To make this factor easier to calculate, the amount that the robot's Joint 1 has to move is used as an approximation to the path length. A lower number is better.

- Figure of merit - a number calculated by the high-level planner that indicates how useful a particular plan is for achieving the system goal of completing sandwiches with minimal wasted ingredients. This is a calculated as the product of the fraction of conveyor travel left for both the ingredient and the base in question, i.e. it is lower when the ingredient and/or base are closer to the ends of their conveyors. A lower number is better.

The plan selector combines these factors by creating a weighted sum, each with an appropriate sign to take account of whether lower or higher numbers are better. Note that in the remainder of this report an appropriate sign is automatically applied to all the weights of these factors so that they can always be considered "better" for more positive or least negative values, even if the sign is not explicitly mentioned.

Unfortunately, the use of this weighted sum leads to the problem of how the correct weighting for these factors should be chosen since it is to be expected that there is an optimal set of weightings that should be used. Trials using simulated annealing techniques to determine the weightings proved to be ineffectual since it was impractical to run the system enough times, and for a long enough duration each time, to arrive at optimum weightings. The shortest practical run of 240 simulated seconds (which resulted in the number of successfully placed ingredients being 100 or less) took around 60 seconds of real time to run on the fastest available computer, which contained an Intel Xeon E5-2650 octa-core 2.60GHz processor. Shorter simulation run times would have resulted in large errors due to starting and stopping effects. Therefore, running the simulation the many thousands of times required to locate the optimum values using simulated annealing, or any similar technique, for each of the scenarios considered would have been impractical, especially since the high levels of noise present in the measurements, when these short run times are used, caused a large number of false local maxima in the function being optimised.

It was decided that a more practical approach was to plot the performance of the system against weightings and inspection to identify the optimum weightings. A typical result from such a test is shown in Figure 40. Note that the two horizontal axes on this graph are pseudo-logarithmic. What is meant by pseudo-logarithmic is that for positive values, the logarithm of the value is used and for negative values, the negative of the logarithm of the negated value is used. A problem with this approach is that it is unable to represent 0. To overcome this, the positive and negative logarithmic ranges are chosen to go from 0.0001 to 1,000,000 and the zero value is made a special case and is mapped directly to 0.

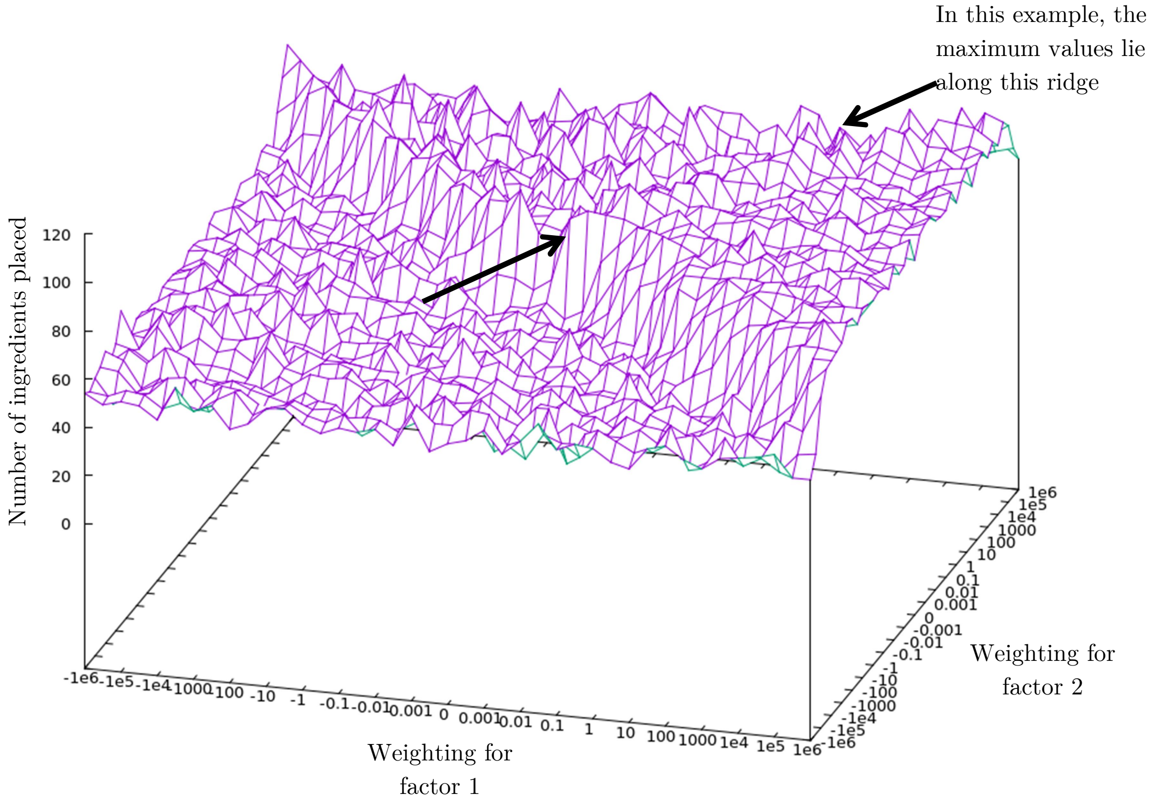


Figure 40: Typical "fitness landscape" of no. of ingredients placed against two weighting factors.

Since these plots can only show the fitness value against two variables in a 3D plot, the third weight has to be fixed. The value used for the fixed third weight will always be shown on the graph. If the third weighting factor is non-zero and has an effect on performance, this also has the advantage of establishing an overall scale factor for the graphs. Without this non-zero fixed weight, any set of weights with the same relative values would yield the same results.

It can be seen that the data in Figure 40 is very noisy and has low resolution (41 x 41 points). The noise occurs due to the short duration of each run (240 seconds) resulting in only a relatively small number of ingredients placed (approximately 100), leading to quantisation errors and errors due to exact the starting and stopping conditions. The quality of the graph could be improved by increasing the number of sample points and running multiple trials and averaging the results, or making each simulation run last longer so that the signal to noise ratio is increased. However, since running the simulation is computationally intensive, it is impractical to do either of these when many data points are required to precisely locate a maximum value.

However, even with the noisy data, it can be seen by inspection of the example above that the approximate region where the optimum occurs lies along the indicated ridge. This region can then be explored in more detail to determine a more precise optimum, although it is clear from the graphs that high precision in the weightings is not necessary.

4.4.4 Optimising the plan selector weightings in simulation

As described in Section 4.3.4, the plan selector has been adopted in order to increase the efficiency of the system when operating in the presence of human co-workers that can cause the robot to stop due to their proximity.

The first task is to arrive at the correct set of weightings for the three factors that affect the plan choosing algorithm. To do this the simulator was used extensively since it would be completely impractical to perform the required number of trials using the Grail testbed.

As was mentioned in Section 4.4.3, it was initially hoped that an automated optimisation method could be applied to the problem to determine the optimum weightings. To achieve this an optimiser program was developed which would modify the settings used during the simulation run as required (all the settings are stored in an xml file and so are easy to modify) and then run the GRAIL Robot Control System in a completely simulated environment for a period of time. To aid this, the GRAIL Robot Control System was given additional features including the ability to run for a pre-determined period of time and then exit, and the ability to count the occurrence of certain high-level events such as ingredients placed and sandwiches completed.

It was decided that the number of ingredient placed on bases in a fixed period of time was an appropriate measure of the performance of the system since this number will be affected by both the time delay due to any stoppages and by any ingredients that are rejected when the robot stops. The number obtained could then be compared with a similar number gathered when there are no co-workers present, and therefore no stoppages, to determine the efficiency (loss) of the system.

Although, as mentioned in Section 4.4.3, simulated annealing was tested as an automatic method for determining the optimum weightings failed, it was realised that a simpler method could yield the desired results. Inspection of the “fitness landscape” when the performance was plotted against the one or more weight values proved to be both simple to implement and more effective. The first baseline test involved plotting the number of ingredients placed against H-R proximity weighting and figure of merit weighting with the path length weighting set to 1.0. The simulated co-workers were placed 5m away from the robot so that they had no effect on performance of the system. The results shown in Figure 41 were obtained.

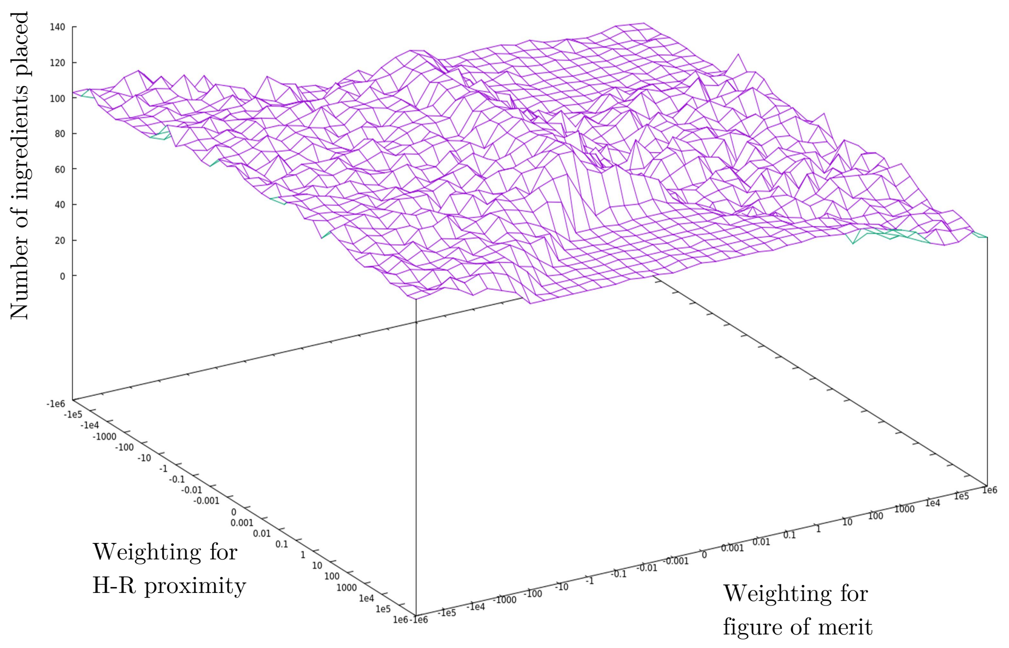


Figure 41: Graph of number of ingredients placed against H-R proximity weighting and figure of merit weighting with path length weighting = 1.0.

This graph shows a number of interesting features. First, the ridge in the centre of the plot occurs when the weighting for path length (which is fixed at 1.0) is dominant. In this region, maximum performance (number of ingredients placed) is obtained.

Secondly, the region with negative weighting for the figure of merit shows better performance than the region with positive figure of merit. This is unexpected since the figure of merit was intended to ensure that ingredients and bases are not wasted when they leave the ends of the conveyors. Although this potentially reduces wastage, the graph demonstrates that the figure of merit metric is detrimental to overall efficiency in terms of the number of ingredients placed.

Since the figure of merit metric has higher values for pick locations and place locations that are closer to the ends of the conveyors, it can be seen in Figure 42 that if the robot preferred to pick and place in these regions, then the robot moves would be quite long and so efficiency, in terms of ingredients placed, would be reduced. The benefit gained from higher figures of merit, less wasted ingredients and bases, is not measured by the fitness metric chosen. For this reason, it was decided to set the weighting for figure of merit to 0.0 (i.e. the plan selector does not choose plans based on the value of figure of merit) for the remainder of the trials.

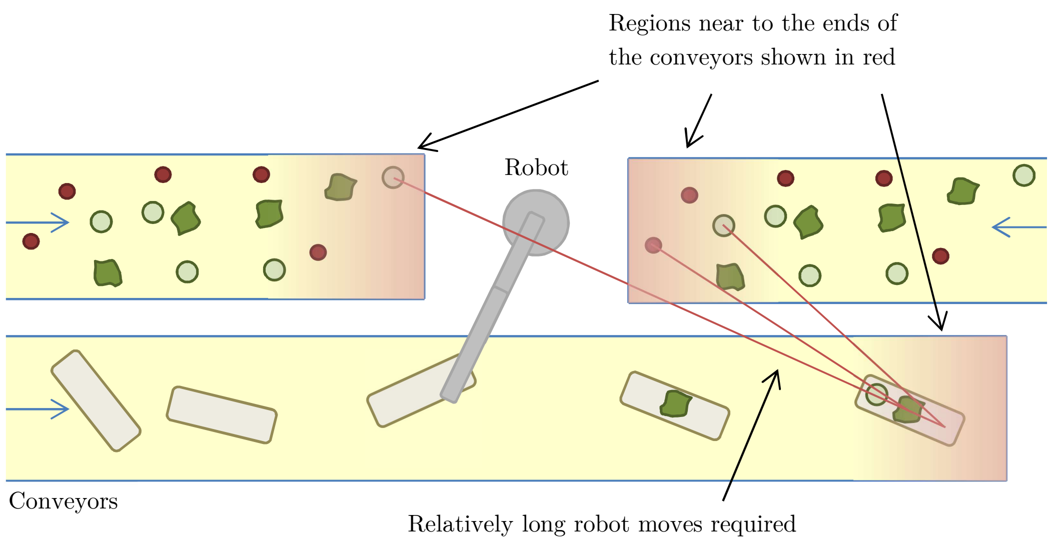

Regions near to the ends of the conveyors shown in red
Robot
Conveyors
Relatively long robot moves required

Figure 42: Depiction of picking and placing regions with high figure of merit factor.

Initially, to maximise the effect that the H-R proximity was having on the results, an idealised version of the system was trialled using a “perfect” prediction of the motion of the co-workers using the methods described in Section 4.1.2. Of course it is not possible to get close to this performance in the real world, but it serves as a good initial test to establish an upper bound on the results that can be expected.

Figure 43 shows the results that were obtained. Note that the two horizontal axes are now path length weighting and H-R proximity weighting since the figure of merit weighting is fixed at 0.0. The weight for the H-R proximity factor was varied along the approximately left-right axis and the weight for the path length factor was varied along the approximately front-back axis.

As expected, the performance of the system is shown to be very good in the region in which the H-R proximity weighting is dominant (shaded blue in Figure 43). The performance is also good in the region that is shaded green in Figure 43, which represents a condition in which path length is dominant.

It can be seen that there is a small suggestion of a maximum height ridge that runs diagonally from the centre to the back, right-hand corner of the graph (shaded red in Figure 43). A typical point on this ridge corresponds to the case where the weighting for the H-R proximity is approximately 10.0 and the weighting for path length is 1.0. Due to the fact that the figure of merit weighting was set to 0.0 and so had no effect, any positive multiple of these weightings also produced similar results, which is demonstrated by the fact that the ridge has constant height. This region is where the best performance is obtained and is explored further below.

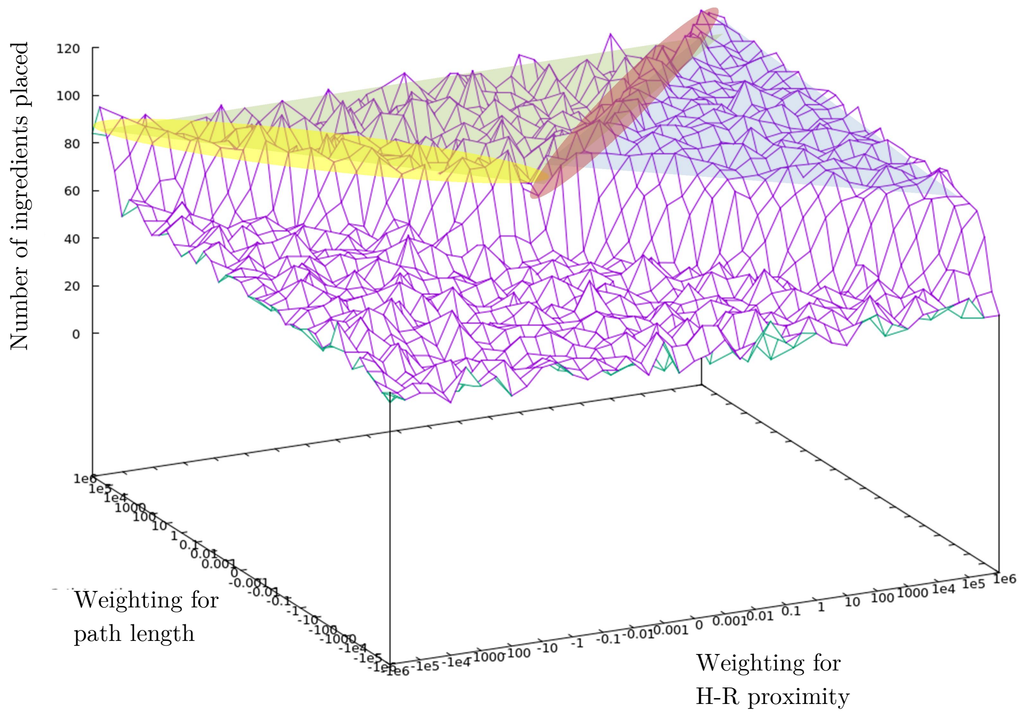


Figure 43: Graph of number of ingredients placed against H-R proximity and path length with figure of merit = 0.0 using perfect predictions.

Interestingly, the performance is also quite good when the weighting for path length is positive and the weighting for H-R proximity it is negative (shown by the yellow shading in Figure 43). This can be explained from a consideration of the geometry of the robot, conveyor and co-worker positions as shown in Figure 44. It can be seen that favouring the regions indicated in red and, at the same time, minimising the path length will generate highly efficient movements of the robot. This is especially true since the robot's speed is mostly determined by joint 1 and this joint will have to move least when picking and placing towards the limits of its reach. The negative H-R proximity weighting tends to keep the robot working at the limits of its reach which is where path length will be generally minimised, whereas using path length alone, although optimum on a pick-by-pick basis, does not constrain the robot to stay in the region that is best for *future* picks and places.

To gain a more detailed insight into the details of what was happening, further trials were done in which the path length weighting was fixed at 1.0 with the figure of merit still set to 0.0. Since this then become a 1-dimensional problem it was practical to run tests at more sample points (81) and make each one have a longer duration (1000 seconds). The results graphed in Figure 45 show the average of 16 separate test runs under these conditions. In the graph the random number generator used to generate ingredient and base positions was seeded with a different number to make the tests suitably independent.

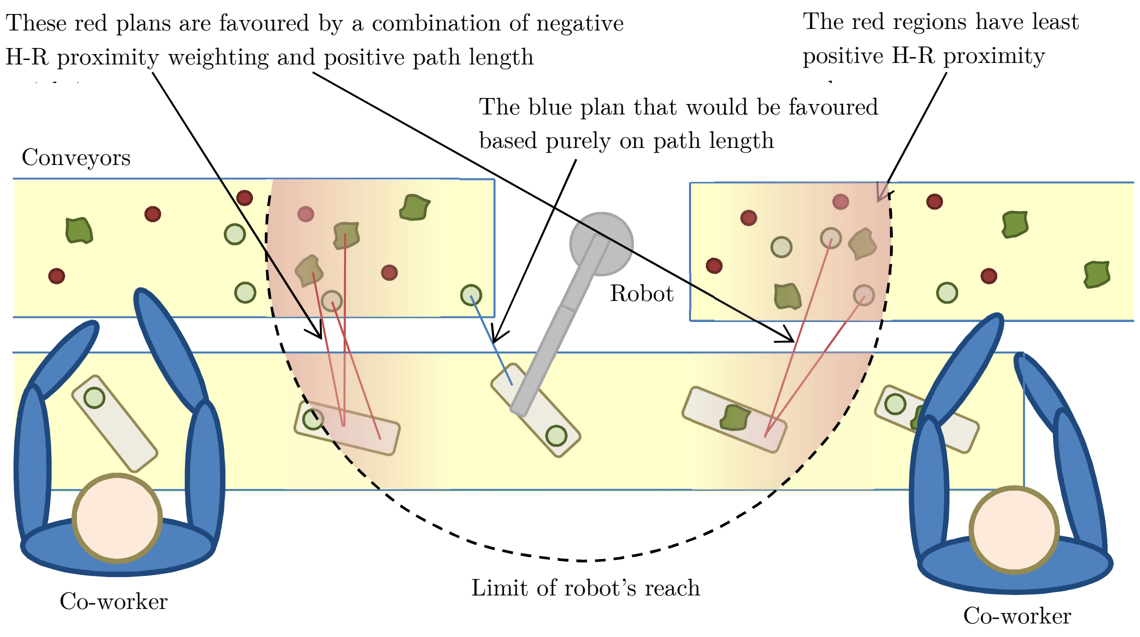

These red plans are favoured by a combination of negative H-R proximity weighting and positive path length
The red regions have least positive H-R proximity
The blue plan that would be favoured based purely on path length
Conveyors
Robot
Co-worker
Limit of robot's reach
Co-worker

Figure 44: Depiction of picking and placing regions with least positive H-R proximity factor.

It can be seen from the graph that there is a clear optimum value when the H-R proximity weighting is around 30, the path length weighting is 1.0 and the figure of merit weighting is 0.0.

Also of note is the local maximum near to a H-R proximity weighting of around -10.0. This is occurs when the negative H-R proximity weighting and positive path length weighting combine to produce a "super path length" optimisation effect as described in the paragraphs above.

The previous test was then repeated using a realistic human position prediction model (using the Kalman filter to determine joint velocities). This achieved the results shown in Figure 46.

Although not as clear cut as for the "perfect prediction" case, there is a hint of a peak along the same diagonal line as before. Again, multiple 1-dimensional runs were performed with the path length weighting set to 1.0 and the figure of merit weightings set to 0.0. The results shown in Figure 47 were obtained by averaging 16 such runs with different random number generator seeds.

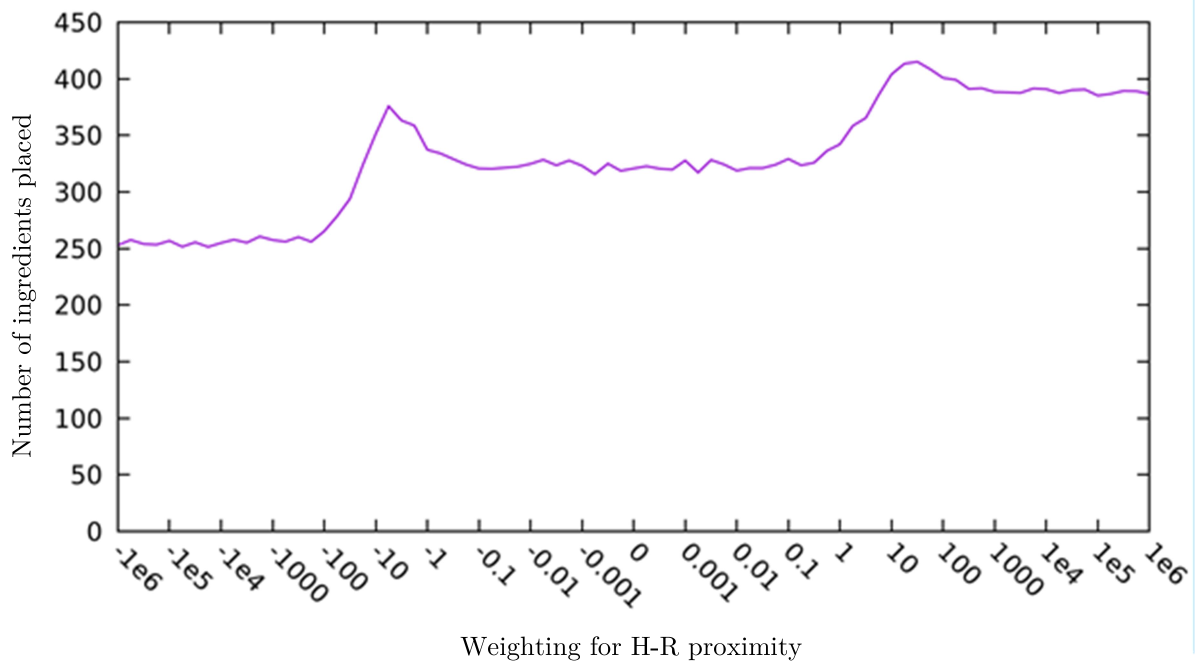


Figure 45: Graph of number of ingredients placed against H-R proximity with path length set to 1.0 and figure of merit set to 0.0 and using perfect predictions.

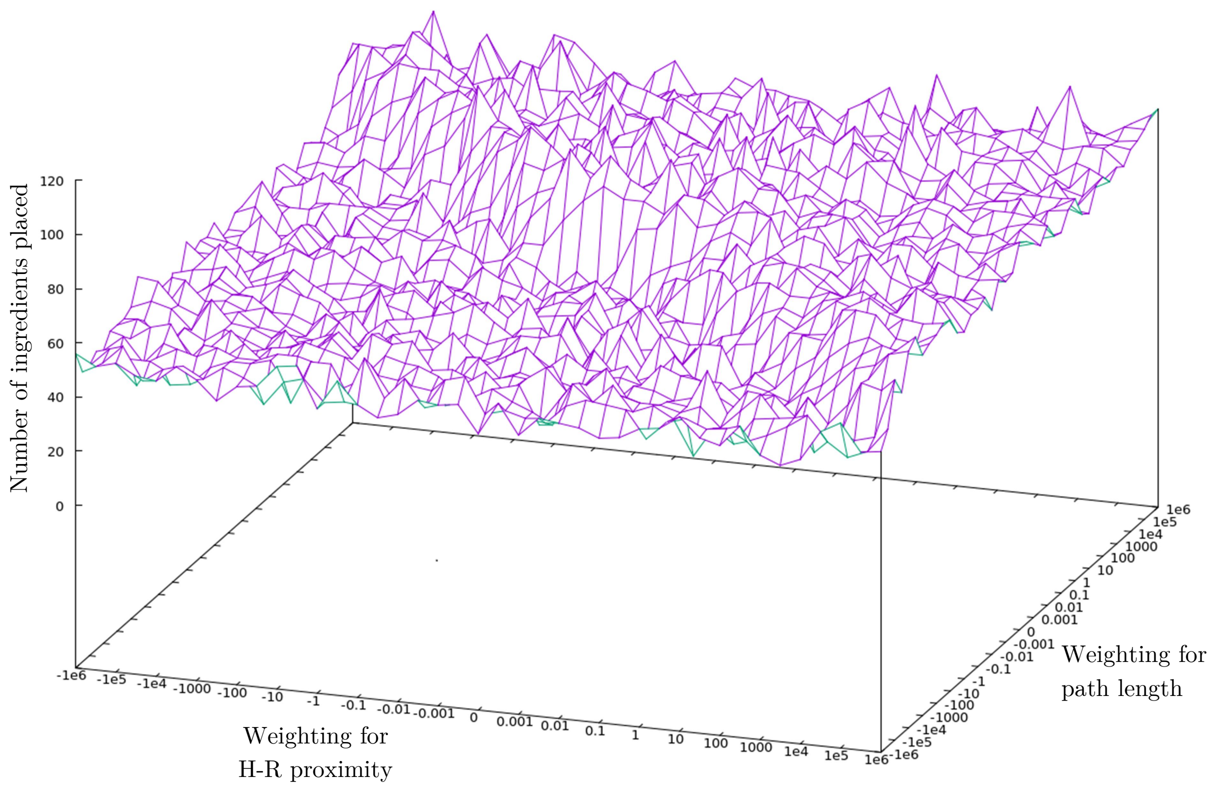


Figure 46: Graph of number of ingredients placed vs H-R proximity and path length with figure of merit = 0.0 using realistic predictions.

Again it can be seen that an optimum result occurs when there is a positive H-R proximity weighting. This time it occurs at H-R proximity weighting of approximately 6. However, in this case the benefit from using the plan selector is much smaller. At the value on the far right of the graph, where the choosing behaviour is dominated by the H-R proximity, the performance is no better than when the weighting for the H-R proximity is set to 0.0. However, at the particular weighting value of around 6, the H-R proximity weighting and path length weighting combine to produce optimum performance.

It can also be seen again that there is a local maximum at approximately -10.0 that is caused for the same reason as described earlier in this section. This time, however, the magnitude of this peak is very nearly the same as the global maximum when H-R proximity weighting is 6.

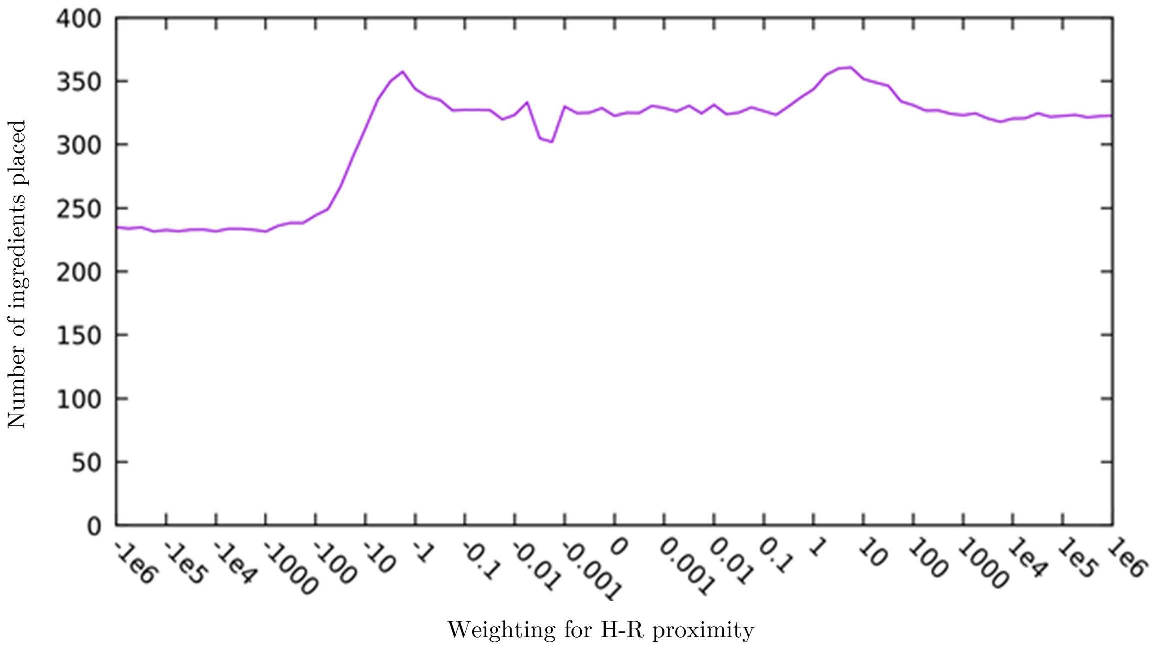


Figure 47: Graph of number of ingredients placed against H-R proximity with path length set to 1.0 and figure of merit set to 0.0 using realistic predictions.

4.4.5 Testing the plan selector performance in simulation

The weighting arrived at in Section 4.4.4 where used to perform a test run of the GRAIL System in which various performance metrics were measured. **Table 3** shows the results that were obtained.

Table 3: Plan selector performance – simulation.

	Placed ingredients	**Number of completed sandwiches**	**Number of rejected ingredients**	**Number of interruptions**	**Number of light curtain breaches**
Without plan selector	397	72	26	95	486
With plan selector and "perfect" predictions	572	86	8	28	491
With plan selector and "real world" predictions	463	52	23	59	495
With plan selector and no predictions	455	57	24	89	497

It can be seen that when the sensor with "perfect" prediction of human motion is used, the plan selector improves the efficiency of the system by nearly 38%. However, when a realistic sensor model is used, the benefits of the plan selector are reduced to 17%. However, the final row in the table shows that there is a 7% gain in efficiency without any modification to behaviour depending on human motion. This is because the plan selector has the additional benefit of choosing more plans with shorter path lengths.

It is interesting to note that changes to the efficiency in terms of number of placed ingredients are not exactly correlated with the number of sandwiches that are completed. In fact, when the system is run without the plan selector active, the second highest number of completed sandwiches is achieved even though the performance, as measured by the number of ingredients placed, is the lowest. When the plan selector is not used, the figure of merit generated by the high-level controller is responsible for selecting which plan to use, and this is optimised to complete as many sandwiches as possible, rather than making many that are incomplete. In practice, the GRAIL System will only be one of many workers on the production line, both human and robotic, and so placing as many ingredients as possible is more important than producing completed sandwiches since downstream workers can do that. If the GRAIL System is placed at the final position on the production line, then optimising for number of ingredients placed will automatically result in maximising completed sandwiches since there will be few locations available in which to place the ingredients and these will, of necessity, result in completed sandwiches.

4.4.6 Testing the plan selector performance in the real world

Experiments were performed using the physical system to demonstrate the performance obtained was similar to the results obtained from running the simulation. As described in Section 4.3.3, limitations of the testbed meant that the right-hand co-worker, light curtain, Kinect sensor, ingredients and bases were simulated.

Although it is harder to get many numerical results, tests were performed in which the system was run both with and without the plan selector active. The results obtained can be seen in **Table 4**. It can be seen from these results that the plan selector is improving the efficiency of the system by approximately 22% (cf. 17% from the simulation) based on number of ingredients placed under the specific conditions tested. Again it can be seen that similar results are obtained even when no predictions are made.

Table 4: Plan selector performance. The numbers in brackets show the results scaled to a 1000 second run – testbed.

	Placed ingredients	Number of completed sandwiches	Number of rejected ingredients	Number of interruptions	Number of light curtain breaches
Without plan selector	81 (324)	13 (52)	11 (44)	27 (108)	105 (420)
With plan selector and “real world” predictions	99 (396)	11 (44)	11 (44)	22 (88)	118 (472)
With plan selector and no predictions	94 (376)	9 (36)	10 (40)	25 (100)	110 (440)

As noted in Section 4.4.5, the number of completed sandwiches does not exactly correlate with the performance results for the performance of the system due to the difference in optimisation criteria between the trials.

4.4.7 Conclusions from testing the plan selector

The results of the previous section have clearly shown that the plan selector increases the efficiency of the GRAIL System when it uses the measured joint velocities of the co-workers to estimate their future movements during a pick and place operation. Some of this improvement, however, is due to the path length-optimisation rather than predicting human motion. However, testing in simulation using a “perfect” prediction of the future movements of the co-workers shows a larger improvement. This indicates that the simplistic prediction methods used in practice, although slightly beneficial, were not very good at predicting the co-workers movements. Future work could investigate whether better prediction methods are available which could approach the performance of the “perfect” sensor more closely.

4.5 Meeting the required safety integrity level

The current implementation of the GRAIL System does not meet high safety integrity level requirements. Meeting these requirements in a development environment is not necessary since different assumptions on the competency of the users can be made and additional protection measures can be used, compared to a production environment. When the GRAIL System is used in a production environment, however, much higher requirements will have to be met.

As first described in Appendix A.4 of Deliverable 5.1[1], the implementation of the GRAIL System is highly constrained by the safety requirements imposed by legislation. The safety-related parts of the GRAIL System are classified as a “protective device” and so must meet the requirements of performance level “d" (PL=d) as defined within EN ISO 13849-1:2008 [3]. Unfortunately, the provable safety of the control algorithms alone is not enough to ensure that the system always operates safely. Additionally, both the hardware and software implementations of the system must be designed to meet the requirements of PL=d.

To meet these requirements, certain safety features must be present in the design. In particular, single faults must not permit an unsafe condition to occur and there must be a means to monitor and detect faults in the safety system before they occur. A first step in achieving this is to identify which parts of the system are parts of the safety system and so must conform to PL=d. Figure 48 shows an overview of the software and hardware architecture of the GRAIL System with the parts that need a high safety integrity level highlighted.

It can be seen that all of the higher level parts system, including the plan selector and associated Kinect sensor, are not safety critical. The plan selector software module and associated Kinect sensor do not need to have a high safety integrity level since safety is still maintained by the verifier if the plan selector or its sensors should malfunction. The only detrimental effect is a reduction in the throughput of the system if the plan selector should fail to select the best plan.

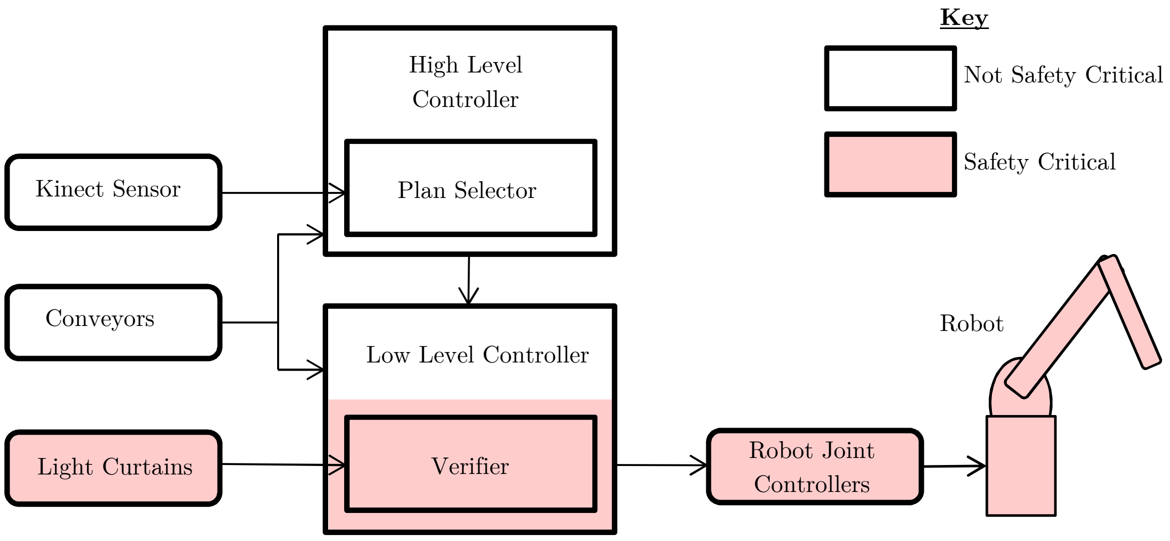


Figure 48: GRAIL System architecture showing safety critical components.

The verifier software module and its associated sensors (light curtains) are responsible for making the robot stop when a hazardous condition occurs. For this reason, they are safety critical and must be implemented in a manner that meets the PL=d safety integrity level.

The robot itself and its joint controllers also need to have high safety integrity level to ensure that the robot always carries out its intended motion. A failure in a position feedback device, for instance, could cause an extreme uncontrolled movement, which could present a hazard.

It is noteworthy that parts of the low-level controller are safety critical and other parts are not. Parts of the software that are responsible for acting on the results produced by the verifier and passing the motion segments which are deemed to be safe by the verifier to the robot joint controller are clearly safety critical, however, other parts of the low-level controller do not affect safety. Conveyor tracking, for instance, is not safety critical since its effects on the robot motion are checked by the verifier and so cannot result in unsafe behaviour.

In general, the required safety integrity level will be met through redundancy and cross checking. Figure 49, which was first presented in “D5.1 – Report on Application Models“ [1], shows how the low-level controller and safety system (the verifier) can be made safe though duplication of functional modules and appropriate cross-checking between the two systems. Redundancy and cross-checking is only effective if there is an assurance that there are no common mode failures between the two systems that could result in an unsafe collision. A common way of achieving this is through the use of independent development teams for each duplicated subsystem with as much diversity as possible in their development processes.

Of course, it is also very important to ensure that the non-safety related parts of the control system do not compromise the safe operation of the critical parts. In practice this means that the non-safety related software must be executed on a separate processor so that, for example, the memory used by the safety critical parts cannot be corrupted. The communication with the safety-critical parts of the system must be rigorously specified in order to avoid any possibility of a fault trickling down into the safety-related parts of the system. The software within the GRAIL System has been written in a modular manner with well-defined and minimal interfaces between its component parts so that this can be achieved. In summary, the implementation of the UnCoVerCPS online verification approach within the GRAIL robot system with the required safety integrity level represents a significant, but not unreasonable, development overhead.

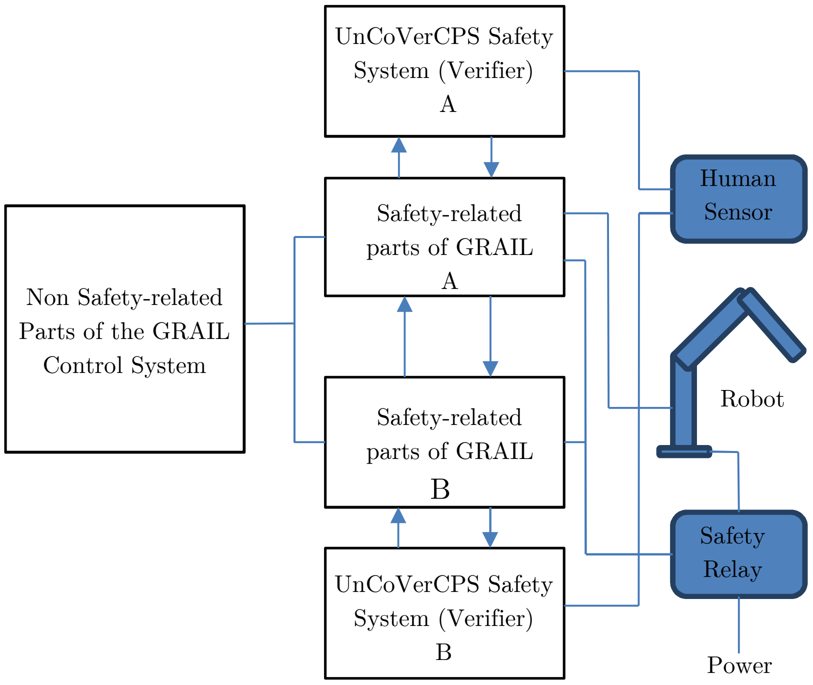

UnCoVerCPS Safety System (Verifier) A
Human Sensor
Safety-related parts of GRAIL A
Non Safety-related Parts of the GRAIL Control System
Robot
Safety-related parts of GRAIL B
Safety Relay
UnCoVerCPS Safety System (Verifier) B
Power

Figure 49: Implementation of the safety-related parts of the control system using duplication and cross-checking.

5 Mobile Robotics Use Case

Within UnCoVerCPS, Bosch introduced the new use case of mobile service robots. In cooperation with Prof. Althoff (TUM), the goal was to evaluate the UnCoVerCPS online verification methods on this use case as well as to perform conformance testing of a pedestrian model.

Regarding the pedestrian model, an abstract pedestrian model was developed which was then tested regarding reachset conformance to a labeled video source of a street scene in Zurich, Switzerland [33]. The pedestrian model was then used to predict pedestrian behavior within the implementation of the UnCoVerCPS online verification method. Here, a ROS simulation was used to evaluate the UnCoVerCPS approach against to other state-of-the-art approaches for three different scenarios. The results showed that, using the UnCoVerCPS online verification method, the robot can maneuver in large and dense groups of people significantly more efficiently than the compared state-of-the-art methods. All details on the performed work can be found in the UnCoVerCPS deliverable D5.2 [2] as well in the joint publication [28].

In addtition to this work, Bosch extended the ROS simulation setup with online conformance monitors for trace and reachset conformance. The goal of these monitors is the use of online reachability computation for checking the plausibility of object classifications from perception algorithms. If an object classified as a walking person shows physical behavior that does not agree with that classification, this information can be used to trigger fallback behavior or improve the classification algorithm.

In particular, to evaluate this use of online reachability analysis, a new class of human behavior was added to the already existing class of pedestrians: joggers. The physical model of new class “jogger” is an extension of the pedestrian model and was again conformance tested against the labeled data of the street scene in Zurich [33]. The new model of a jogger now also captures behavior that was not included within the pedestrian model (i.e., higher accelerations and velocities). In the ROS-based simulation setup, online reachset conformance checking based on precomputed reachable sets was used to classify the physical behavior of pedestrians either as walkers or joggers. This work realizes the important task of also being able to do online conformance monitoring, in order to detect deviating behavior that was not foreseen at the design time of a system.

6 Overall Conclusions

This report described the results that were achieved by applying the methods developed within the UnCoVerCPS project to the four diverse domains of automated driving, human-robot collaboration, mobile robot navigation and smart grids. It also presented results showing how the achievements were demonstrated though experimental testing. Taken together, these results show the potential for the wide application of the UnCoVerCPS approach.

For the automated driving use case, a linear model predictive approach with a decoupled lateral and longitudinal model was successfully demonstrated and provided the capability for successful lane keeping, lane changing and overtaking manoeuvres.

The human-robot use case example demonstrated how the UnCoVerCPS online verification approach was adopted into a practical realisation of a robot control system and offered significant benefits over more traditional safety approaches such as physical barriers or simple systems that cause the robot to perform an emergency stop every time a co-worker is too close.

The mobile robots use case also used the UnCoVerCPS online verification methods and demonstrated increased performance of mobile service robots compared to standard approaches. Results were achieved that successfully demonstrated that, especially in dense environments with large, dense groups of people and other obstacles, the mobile robots were able to reach their destinations considerably faster than would be possible using other approaches. In order to apply the UnCoVerCPS approach, a conformant pedestrian model was identified using CORA and the conformance checking methods developed within earlier WP5 tasks (see Deliverable D5.2 (2)).

Within the smart grid use case a solution is proposed that uses a novel data-driven approach with probabilistic guarantees to address the design of a disturbance compensator that tries to moderate the fluctuations of the energy production due to solar energy. Results from trials were presented which demonstrated that this approach has significant benefits.

These applications, while each utilising differing sub-sets of the full UnCoVerCPS toolbox demonstrate the wide capabilities the UnCoVerCPS approach when appropriately applied. The fact that the methodologies developed within the UnCoVerCPS project have been successfully applied to each of these diverse use cases further demonstrates that the de-verticalisation goals of the UnCoVerCPS project have been met.

Appendix A – Human-Robot Distance Calculation

A.1 Statement of problem.

To allow the GRAIL System to construct sandwiches efficiently, it is necessary to determine the distance between the predicted motion of any nearby co-workers and any particular pick and place plan. Since the robot motion follows trajectories that linearly interpolate both the robot's angle and radial distance during the execution of a plan, the robot arm trajectory comprises a segment of an Archimedes spiral which passes through the two given points in space (in this case, the pick and place locations).

When a co-worker interrupts the light curtain, the human proximity is represented by a plane parallel to the light curtain. The distance between this plane and the light curtain is called the "incursion distance". The distance between the robot and the light curtain (called the "barrier" in the remainder of this Appendix) is then calculated using the method presented below. By subtracting the incursion distance from the robot to barrier distance, any potential overlap (negative result) or clearance (positive result) between the predicted motion of the co-worker and the robot can be determined and used to optimise the choice of plans.

A.2 Initial discussion

There are in general two Archimedes trajectories which pass through any two given points, as shown in Figure A.1:

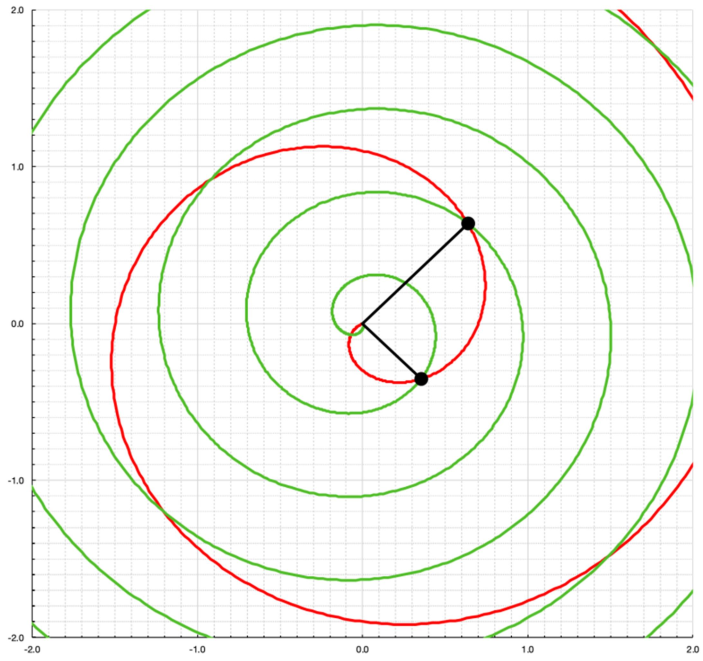


Figure A.1: Two Archimedes spirals passing through two common points.

In practice, we assume that the robot is not capable of executing trajectories for positions exceeding 170° away from the x-axis. This always removes one of the two tracks from further consideration; in the figure above this would eliminate the green track, but not the red track (as the red path between the two endpoints stays between -45 and +45 degrees).

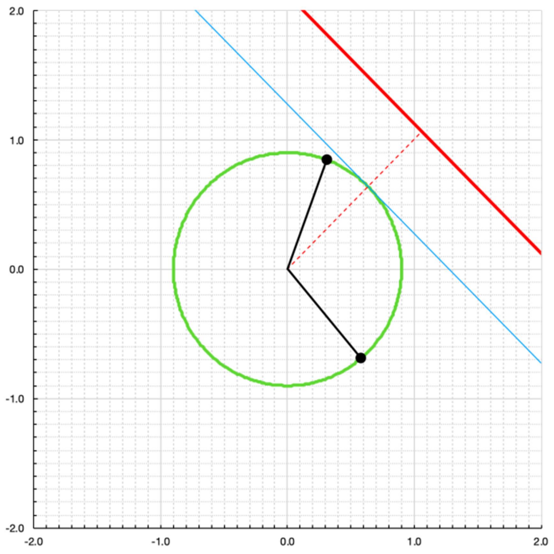


Figure A.2: Circular trajectory.

Figure A.2 shows an example where the radii of the two end-points (in black) are the same so the trajectory (shown in green) is circular. The barrier is shown in red, and the normal line from the origin to the barrier is shown as a dotted red line.

The blue line is parallel to the barrier and indicates the point of closest approach to the barrier. The distance between the blue line and the barrier is the minimum distance required.

It is noted that, at the point of closest approach, the trajectory is running parallel to the barrier (so the end-effector is neither moving towards nor away from the barrier), and the tangent touches at the same point that the dotted red line (the barrier normal) crosses the trajectory line. This is to be expected with a circular track but is not the case with a non-circular track.

For example, in Figure A.3, the (red dotted) barrier normal line crosses over the green trajectory line at a different point to the closest approach point (where the blue line touches). When the robot arm ‘points towards’ the barrier it will not be at the point of closest approach as it is still approaching the barrier. Note however that the point of closest approach still lies along the path between the endpoints in this case.

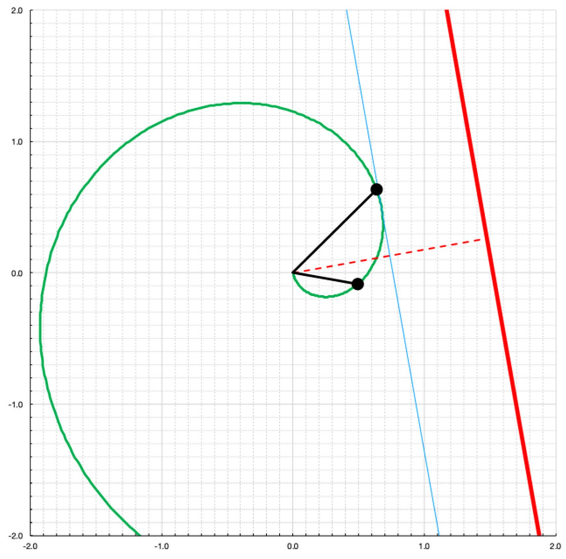


Figure A.3: A non-circular path.

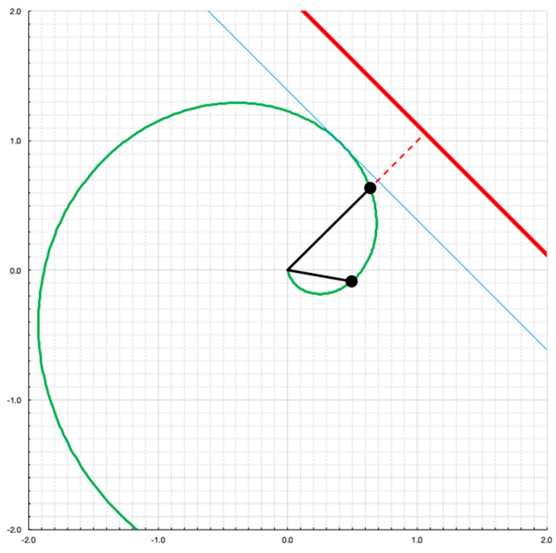


Figure A.4: Touch points outside path.

Similarly, in Figure A.4 the barrier normal line is at the end of the path, but because the end-effector is still approaching the barrier at that point, the point of closest approach of the Archimedes trajectory moves out beyond the path. In this case, the closest point of approach **of the path** to the barrier is the path endpoint itself.

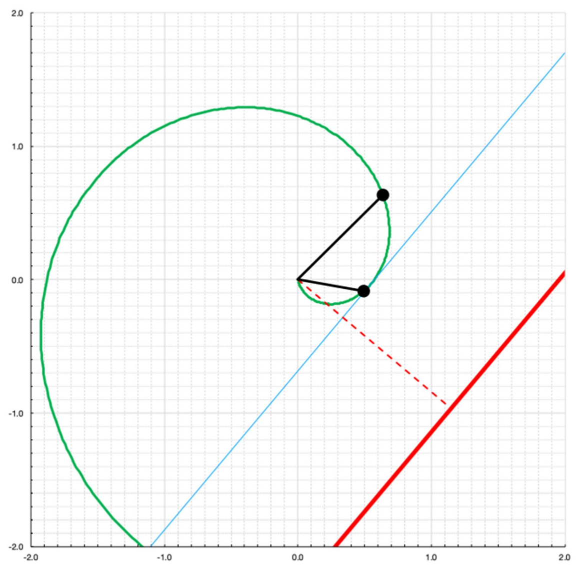


Figure A.5: Touchpoint in path, normal outside path.

Figure A.5 shows another case, where the point of closest approach is within the path, but the normal to the barrier is well outside the path.

Note that in all four scenarios above, the path between the start and end points misses the barrier.

Odd Cases

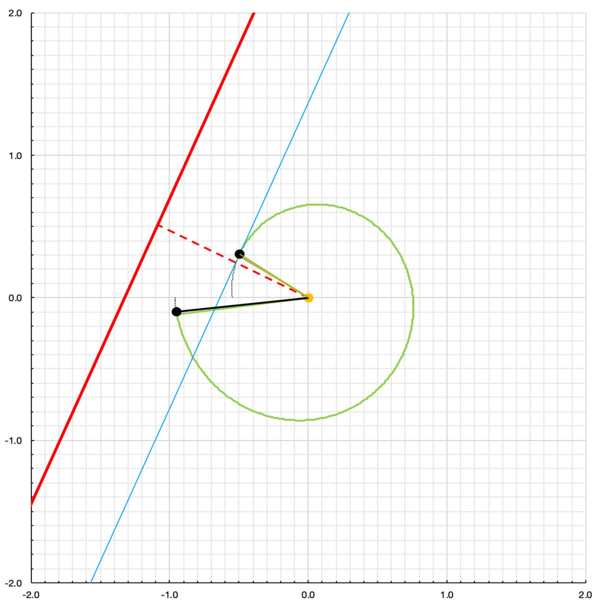


Figure A.6: Case where point of closest approach is not the tangent line.

Figure A,6 shows a case where the tangent point in the trajectory is not the closest point in the trajectory to the barrier. In this case, one end-point is closest (as it happens the robot is at its limit at that point too).

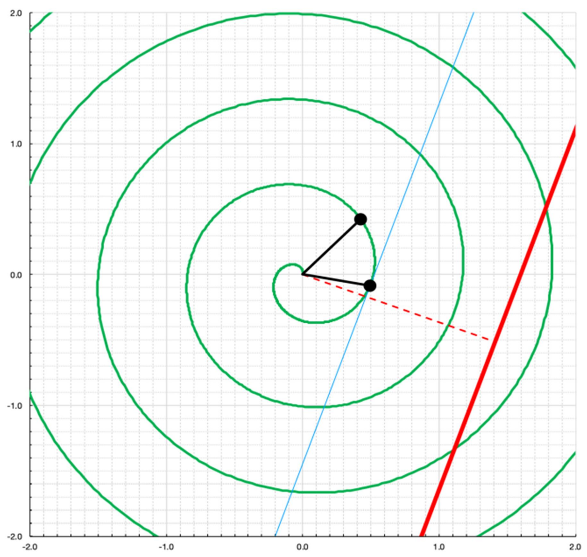


Figure A.7: Multiple cycles of trajectory between robot and barrier.

Also, it is necessary to be careful that the correct part of the Archimedes trajectory is used to calculate the point of closest approach. Figure A.7 shows a case where the path cycles round more than once between the robot and the barrier. It is important to choose the portion of the path parallel to the barrier closest to the robot.

Straight line (radial) trajectories.

In the case of a trajectory which is purely radial (i.e. the polar angles of the endpoints are the same), then the only test that is needed is to compute the distance to the barrier of the endpoints as there is no tangent point. The closest endpoint represents the closest point in the trajectory. If one distance is negative, then the path crosses the barrier.

Therefore, the algorithm to use is:

1. Identify the correct tangent point and if this is between the endpoints. If so identify where it is and how far it is from the barrier.
2. Compare the distance to the barrier of the tangent point with both end-points of the path.
3. Of three options (end points and tangent point), choose the point closest to the barrier.

Finding the tangent point

The first task is to determine whether the tangent point (where the trajectory is parallel to the barrier) lies between the endpoints.

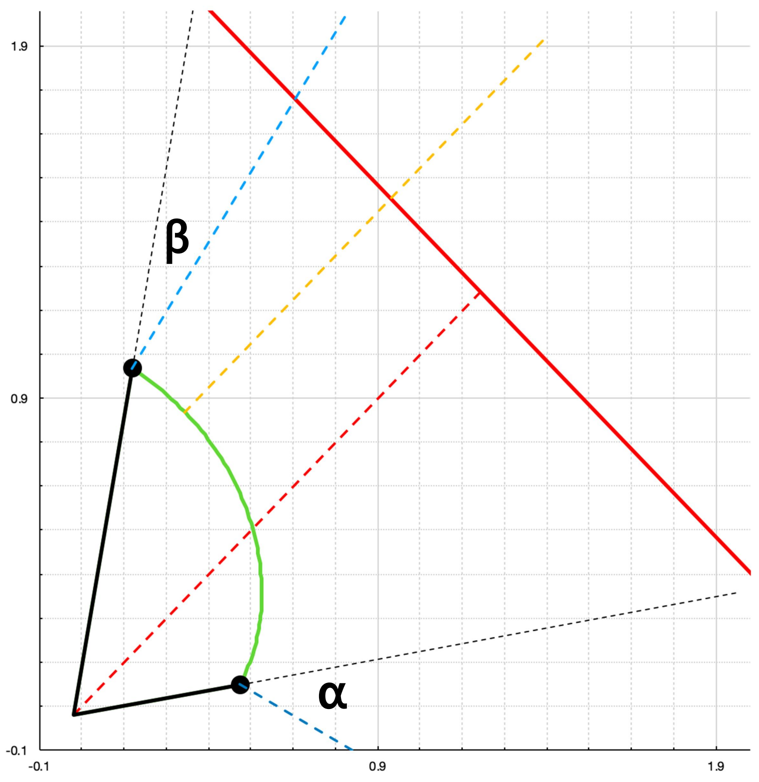


Figure A.8: Various trajectory angles

It is possible to determine whether the point of closest approach is within the trajectory without actually knowing where it lies in the trajectory. Figure A.8 shows a trajectory (in green) under consideration. The start and end points are each characterised by two vectors: - a) The radial directions (in black dotted lines) and b) the normal to the trajectory paths (in dark blue and light blue).

The barrier normal is denoted by the dotted red line. The yellow dotted line denotes the point where the trajectory is parallel to the barrier and so the trajectory normal is parallel to the barrier normal.

The two blue lines denote the bounds of the normals of the trajectory path, while the black dotted lines denote the polar angles of the endpoints (and also the actuation angles of the robot arm at the endpoints).

If the barrier normal angle lies within the bounds of the blue endpoint normal angles, then the tangent point lies between the two endpoints.

The trajectory normals at the endpoints can be computed by subtracting the 'skew angles' **α** and **β** from the polar angles of the trajectory points. These are computed as follows:

An Archimedes trajectory is best described in polar coordinates: -

$$r = a + b.\theta \qquad (A.1)$$

The parameters a and b determine the shape and orientation of the trajectory. The radius r is then determined as a function of θ. Given two points through which the path must pass (r0, θ0) and (r1, θ1), then

$$r0 = a + b.\theta 0 \text{ and} \qquad (A.2)$$

$$r1 = a + b.\theta 1$$

which leads to

$$b = (r1 - r0)/(\theta 1 - \theta 0) \qquad (A.3)$$

and

$$a = r0 - b.\theta 0 = r0 - \theta 0(r1-r0)/(\theta 1-\theta 0) \qquad (A.4)$$

differentiating equation (A.1)

$$dr/d\theta = b \qquad (A.5)$$

Suppose we move along the trajectory by distance dθ. Then, the change in radial distance is

dr = b.dθ (from equation (A.5)). The change in angular distance is r.dθ (when dθ is expressed in radians, only). This can be viewed as a right-angle triangle where the adjacent side is of length r.dθ and the opposite side is of length b * dθ. The tangent of the angle is then (b.dθ) / (r.dθ) = b/r. Therefore the 'skew' angle is arctan(b/r); this is the angle **α** or **β** in Figure A.8.

$$\text{skew angle} = \arctan(b/r) \qquad (A.6)$$

In the example shown in figure 8, the polar coordinates of r0 are (0.5, 10°) and point r1 (1.0, 80°).

Thus, b=0.4091 (note: this requires θ1 - θ0 to be expressed in radians).

The skew angle **α** = arctan(b/0.5) = 39.3° and **β** = arctan(b/1.0) = 22.25°.

Therefore:

the angle of the r1 line is 80° - 22.25° = 57.75°

the angle of the r0 line = 10° - 39.3° = -29.3°

The angle of the barrier normal line is 45°. This lies between the limits 57.75° and -29.3°, so we can deduce that the tangent line lies between the endpoints (although as yet we do not know exactly where).

Initial computation of tangent line position.

The position of the tangent point can be estimated using linear interpolation.

Set Φ as the polar angle of the tangent point, and Γ as the polar angle of the barrier normal.

Set $\Theta = \theta 1 - \boldsymbol{\beta}$ = the angle of the trajectory normal at r1 (Θ is the angle of the light blue line in Figure A.8).

Set $\Omega = \theta 0 - \boldsymbol{\alpha}$ = the angle of the trajectory normal at r0 (Ω is the angle of the dark blue line in Figure A.8).

Then the tangent point is in the trajectory if $\Theta >= \Gamma >= \Omega$.

Using linear interpolation, assume

$$(\Gamma-\Omega)/(\Theta-\Omega) = (\Phi-\theta 0)/(\theta 1-\theta 0) \tag{A.7}$$

For example, if the barrier angle Γ is a third of the angular distance between the endpoint trajectory normals Θ and Ω, then set as an estimate Φ = a third of the angular distance between $\theta 1$ and $\theta 0$.

This leads to

$$\Phi \approx \theta 0 + [(\Gamma-\Omega)(\theta 1 - \theta 0)]/(\Theta-\Omega) \tag{A.8}$$

For the example in Figure A.8 above this gives

$\Theta = 57.75°$

$\Omega = -29.3°$

$\Gamma = 45°$

$\theta 1 = 80°$

$\theta 0 = 10°$

$\Phi \approx 10° + [(45° - -29.3)*(80° - 10°)]/(57.75° - -29.3°)$

$= 10° + 59.74° = 69.74°$

This compares well with the actual angle of 68.95°. The estimated distance to the barrier is very accurate using the estimated value for Φ, because the trajectory near this point is very nearly parallel to the barrier and so distance to the barrier is insensitive to small changes in position of the manipulator along the track.

Refining the tangent line position

Once an estimate of the tangent position has been determined, a final (accurate) figure can be obtained by Newton-Raphson iteration, as follows:

The tangent position will be found when $\Phi = \Gamma$.

This can be expressed as

$$\Phi - \Gamma = 0 \tag{A.9}$$

or

$$f(\theta) = \theta - \arctan(b/r) - \Gamma = 0$$

but $r = a + b\theta$, so

$$f(\theta) = \theta - \arctan(b/[a + b\theta]) - \Gamma = 0 \tag{A.10}$$

differentiating wrt θ

$$\begin{aligned} f'(\theta) = df/d\theta &= 1 + b^2/(r^2 + b^2) \\ &= [(r^2 + b^2) + b^2]/(r^2 + b^2) \\ &= (r^2 + 2b^2)/(r^2 + b^2) \end{aligned} \tag{A.11}$$

Using the NR iteration method

$$\theta_{n+1} = \theta_n - f(\theta)/f'(\theta) \tag{A.12}$$

then

$$\begin{aligned} &\theta_{n+1} = (\theta_n - \arctan(b/r_n) - \Gamma)(r_n^2 + b^2)/(r_n^2 + 2{*}b^2), \\ &\text{where } r_n = a + b\theta_n \end{aligned} \tag{A.13}$$

In the example above, the Newton-Raphson iteration converges in one step to the answer $\theta = 68.9558°$ (compared with a start estimate of 69.74°). The initial estimate of distance to the barrier is 0.89625 units, while the final distance is 0.893921 units, a difference of 0.0023.

Computing distance to the barrier from any given point

The computation of the distance to the barrier for a given point is best done in Cartesian coordinates, using scalar products.

Given a point D and a point B and a barrier normal **Ñ**, where

D is the point whose distance to the barrier is needed

Ñ is the barrier normal unit vector.

B is the point on the barrier which is closest to the origin - i.e. the point at which the normal vector crosses the barrier.

Then the distance d is determined as follows:

$$d = (D\text{-}B).\tilde{\mathbf{N}} \tag{A.14}$$

if d is negative or zero, then the point D is on the same (safe) side of the barrier as the robot (or origin). Otherwise it is not.

References

[1] UnCoVerCPS Deliverable D5.1: Report on application models. Technical report.

[2] UnCoVerCPS Deliverable D5.2: Report on conformance testing of application models. Technical report.

[3] *EN ISO 13849-1:2008*. Brussels: CEN, 2008.

[4] *EN ISO 13855:2010*. Brussels: CEN, 2010.

[5] T. Alamo, R. Tempo, A. Luque, and D. R. Ramirez. Randomized methods for design of uncertain systems: Sample complexity and sequential algorithms. *Automatica*, 52:160–172, 2015.

[6] M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. In *IEEE Transactions On Robotics*, volume 30, pages 903 – 918. IEEE, August 2014. DOI: 10.1109/TRO.2014.2312453.

[7] H. Bai, J. Shen, L. Wei, and Z. Feng. Accelerated lane-changing trajectory planning of automated vehicles with vehicle-to-vehicle collaboration. *Journal of Advanced Transportation*, 2017, 2017.

[8] S. Bittanti, M. C. Campi, and M. Prandini. Adaptation and the effort needed to adapt. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 5733–5737, Dec 2009.

[9] C. Brocchini, A. Falsone, G. Manganini, O. Holub, and M. Prandini. A chance-constrained approach to the quantized control of a heat ventilation and air conditioning system with prioritized constraints. In *Proceedings of the 22nd International Symposium on Mathematical Theory of Networks and Systems (MTNS 2016), Minneapolis, Minnesota, USA*, pages 137–144, July 2016.

[10] G. Calafiore and M. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

[11] G. Calafiore and M. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, 2006.

[12] M. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2):257–280, 2011.

[13] M. Campi, S. Garatti, and M. Prandini. The scenario approach for systems and control design. *Annual Reviews in Control*, 33(2):149–157, 2009.

[14] H. Chen, T. N. Cong, W. Yang, C. Tan, Y. Li, and Y. Ding. Progress in electrical energy storage system: A critical review. *Progress in Natural Science*, 19(3):291–312, 2009.

[15] J. Copper, A. Sproul, and S. Jarnason. Photovoltaic (PV) performance modelling in the absence of onsite measured plane of array irradiance (POA) and module temperature. *Renewable Energy*, 86:760–769, 2016.

[16] L. Deori, S. Garatti, and M. Prandini. A randomized relaxation method to ensure feasibility in stochastic control of linear systems subject to state and input constraints. *Automatica*, 2018. Submitted. Online preprint arXiv:1610.06315.

[17] S. Dubey, J. N. Sarvaiya, and B. Seshadri. Temperature Dependent Photovoltaic (PV) Efficiency and Its Effect on PV Production in the World – A Review. *Energy Procedia*, 33:311–321, 2013.

[18] A. Falsone, L. Deori, D. Ioli, S. Garatti, and M. Prandini. Optimal disturbance compensation for constrained linear systems operating in stationary conditions: a scenario-based approach. *Automatica*, 2018. under review.

[19] D. González, J. Pérez, V. Milanés, and F. Nashashibi. A review of motion planning techniques for automated vehicles. In *Transactions on Intelligent Transportation Systems*, volume 17(4), pages 1135 – 1145. IEEE, April 2016. DOI: 10.1109/TITS.2015.2498841.

[20] D. Heß, C. L´oper, and T. Hesse. Safe cooperation of automated vehicles. *the AAET Automatisiertes & Vernetztes Fahren*, pages 309–334, 2017.

[21] D. Ioli, A. Falsone, M. Hartung, A. Busboom, and M. Prandini. A smart-grid energy management problem for data-driven design with probabilistic reachability guarantees. In *Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH 2017)*, 2017.

[22] L. Joerissen, J. Garche, C. Fabjan, and G. Tomazic. Possible use of vanadium redox-flow batteries for energy storage in small grids and stand-alone photovoltaic systems. *Journal of Power Sources*, 127(1):98–104, 2004.

[23] N. Kalra and S. M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? In *Transportation Research Part A: Policy and Practice*, volume 94, pages 182 – 193. ELSEVIER, December 2016. DOI: 10.1016/j.tra.2016.09.010.

[24] A. Khodayari, A. Ghaffari, S. Ameli, and J. Flahatgar. A historical review on lateral and longitudinal control of autonomous vehicle motions. In *2nd International Conference onMechanical and Electrical Technology (ICMET)*, pages 421 – 429. IEEE, September 2010. DOI: 10.1109/ICMET.2010.5598396.

[25] S. Krauter and A. Preiss. Comparison of module temperature measurement methods. In *2009 34th IEEE Photovoltaic Specialists Conference (PVSC)*, pages 000333–000338, June 2009.

[26] H. Kress-Gazit and G. J. Pappas. Automatically synthesizing a planning and control subsystem for the darpa urban challenge. In *4th Conference on Automation Science and Engineering*, page 766 – 771. IEEE, August 2008. DOI: 10.1109/COASE.2008.4626549.

[27] R. Lattarulo, D. Heß, and J. Perez. A linear model predictive planning approach for overtaking manoeuvres under possible collision circumstances. *IEEE Intelligent Vehicles Symposium (IV)*, pages 1340 – 1345, 2018.

[28] S. B. Liu, M. Althoff, C. Heinzemann, H. Roehm, J. Oehlerking, and I. Luetkebohle. Motion verification for robots in human environments based on kinematic pedestrian models. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*, 2017.

[29] S. M. Loos, A. Platzer, and L. Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In *International Symposium on Formal Methods*, pages 42 – 56. Springer, Berlin, 2011. DOI: 10.1007/978-3-642-21437-0_6.

[30] J. Maciejowski. *Predictive Control with Constraints.* Prentice Hall, 2002.

[31] J. Maclay, J. Brouwer, and G. Samuelsen. Dynamic modeling of hybrid energy storage systems coupled to photovoltaic generation in residential applications. *Power Sources*, 163:916–925, 2007.

[32] U. D. of Transportation. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. In *Traffic Safety Facts Crash Stats*, pages 1 – 2. NHTSA, February 2015.

[33] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: modeling social behavior for multi-target tracking. In *Proc. of ICCV*, pages 261–268, 2009.

[34] M. Prandini, S. Garatti, and R. Vignali. Performance assessment and design of abstracted models for stochastic hybrid systems through a randomized approach. *Automatica*, 50(11):2852–2860, 2014.

[35] A. Prèkopa. *Stochastic Programming.* Kluwer, Boston, MA, 1995.

[36] A. Prèkopa. Probabilistic programming. In A. Ruszczyǹski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, London, UK, 2003. Elsevier.

[37] E. Skoplaki and J. Palyvos. On the temperature dependence of photovoltaic module electrical performance: A review of efficiency/power correlations. *Solar Energy*, 83(5):614–624, 2009.

[38] J. K. Tonui and Y. Tripanagnostopoulos. Air-cooled PV/T solar collectors with low cost performance improvements. *Solar Energy*, 81:498–511, Apr. 2007.