



**Sustainable and reliable robotics for part handling in manufacturing**

**Project no.:** 610917  
**Project full title:** Sustainable and reliable robotics for part handling in manufacturing  
**Project Acronym:** STAMINA  
**Deliverable no.:** D1.3.8  
**Title of the deliverable:** STAMINA test and evaluation report M41

**Contractual Date of Delivery to the CEC:** 28.02.2017  
**Actual Date of Delivery to the CEC:** 17.03.2017  
**Organisation name of lead contractor for this deliverable:** Peugeot Citroën Automobiles S.A.  
**Author(s):** Arnaud Chazoule, Volker Krüger, German Martin, Alexander Schiotka, Lazaros Nalpantidis, Cesar Toscano, Ron Petrick, Matthew Crosby  
**Participants(s):** P01, P02, P04, P05, P06, P07  
**Work package contributing to the deliverable:** WP1  
**Nature:** R  
**Version:** 1.0  
**Total number of pages:** 16  
**Start date of project:** 01.10.2013  
**Duration:** 42 months – 31.03.2017

**This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610917**

**Dissemination Level**

<b>PU</b>	Public	<b>X</b>
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	

**Abstract:**

This deliverable provides a report on the tests performed during the third test sprint. This performance report will provide a structured feedback to the following R&D phase and prepare the next test sprint.

## Document History

<b>Version</b>	<b>Date</b>	<b>Author (Unit)</b>	<b>Description</b>
0.1	10.03.2017	V. Krueger	Complete overhaul
0.2	12.03.2017	A. Chazoule	Some minor modifications
1.0	15.03.2017	V. Krueger	Final version

## Table of Contents

<b>Executive Summary</b> .....	<b>4</b>
<b>1 Introduction: Objectives of the fourth test sprint</b> .....	<b>5</b>
<b>2 Description of the test sprint environment</b> .....	<b>5</b>
<b>3 Novel System Components</b> .....	<b>8</b>
3.1.1 Automatic Camera Calibration .....	8
<b>4 Storyboards used during the test sprint.</b> .....	<b>8</b>
<b>5 Experimental Results</b> .....	<b>10</b>
5.1 Baseline missions .....	10
5.2 Startup in the morning and installing a new robot .....	11
5.3 Complex missions .....	11
5.4 Testing the Robot skills .....	13
5.5 Testing skill primitives.....	14
<b>6 Conclusions</b> .....	<b>15</b>
<b>7 Appendix</b> .....	<b>16</b>

## Executive Summary

The agile development methodology applied in STAMINA relies on iterative and incremental test sprints for managing the integration, testing and structured feedback to RTD. Four test cycles have been planned during which all partners will be focused on the test to provide flawless runs and sustainable feedback to the R&D partners.

The present document is the **test and evaluation report for the fourth and final test cycle**. At this final stage, the focus lies on the generalization and combination of the previous developments and on thoroughly testing the system for robustness towards disturbances.

The document is decomposed as follows: the reader will find a general description of the test sprint organization in Section 1. A description of the test sprint environment is provided in Section 2. Section 3 discusses novel components introduced into the system for this final test sprint. Section 4 introduces several story boards which describe the situations, or *stories* in which the robot finds itself. These are typical and realistic situations from the shop floor, and the robot must be able to deal with them. The story boards describe the experiments we have done. Section 5 provide an exhaustive discussion of our extensive results. We conclude with final comments in Section 6.

*Note that this document is identical to Deliverable D1.3.4 (CO, Confidential version).*

## 1 Introduction: Objectives of the fourth test sprint

The fourth test sprint of the STAMINA project took place from M38 to M40. The overall target of this test sprint according to Task 1.3 in the DOW is as follows:

*On a fourth testing cycles test the focus will be on the evaluation of the complete system namely through usability testing. These usability tests will target both the end users, which include blue-collars, workers and forklift drivers but also the systems integrator. The evaluation of the setup time and the engineering efforts required to setup and run the STAMINA system constitute an important foundation for the future exploitation of the project.*

In this test sprint, we have for the second time tested the *complete* system in the relevant environment. Already last time, the tests were focused on testing the execution of complete missions and on generating large amounts of statistics about detectable, undetectable and predictable errors. During the third test sprint a large number of shortcomings became apparent in all parts of the system that impacted the overall system performance. The shortcomings were discussed in Section 7 of D1.3.7.

To summarize the results from the third test spring:

- The complete system suffered robustness. Once we figured out how the system *ticked*, we were able to run repeatedly 3 and 4 part missions. However, for individuals unfamiliar with the system, it was impossible to run the system reliably.
- We were not able to run 5 part missions due to lack of robustness
- Various skills were not able to report errors that could be used by the planner for replanning
- Various skills suffered from undetected fails.

These shortcomings were specifically addressed and solved.

In this test sprint, we have repeatedly completed missions of different complexity. In addition, we have deliberately induced errors and disturbances to the environment to test the reaction of the robot.

The report is structured as follows: In Sect. 2, we give an overview of the test sprint environment at PSA. The environment is exactly the same as for the third test sprint. Section 3 summarizes novel system components of the Stamina system. Section 3 summarizes the test story boards used during our experiments. Section 4 summarizes the experimental results.

## 2 Description of the test sprint environment

PSA has created a new replica of a small logistics kitting zone (see picture below) with containers and shelves for testing the basic robot skills (navigation, part localization, picking, placing). Figure 1 shows four isles with a number of pallets and shelves.



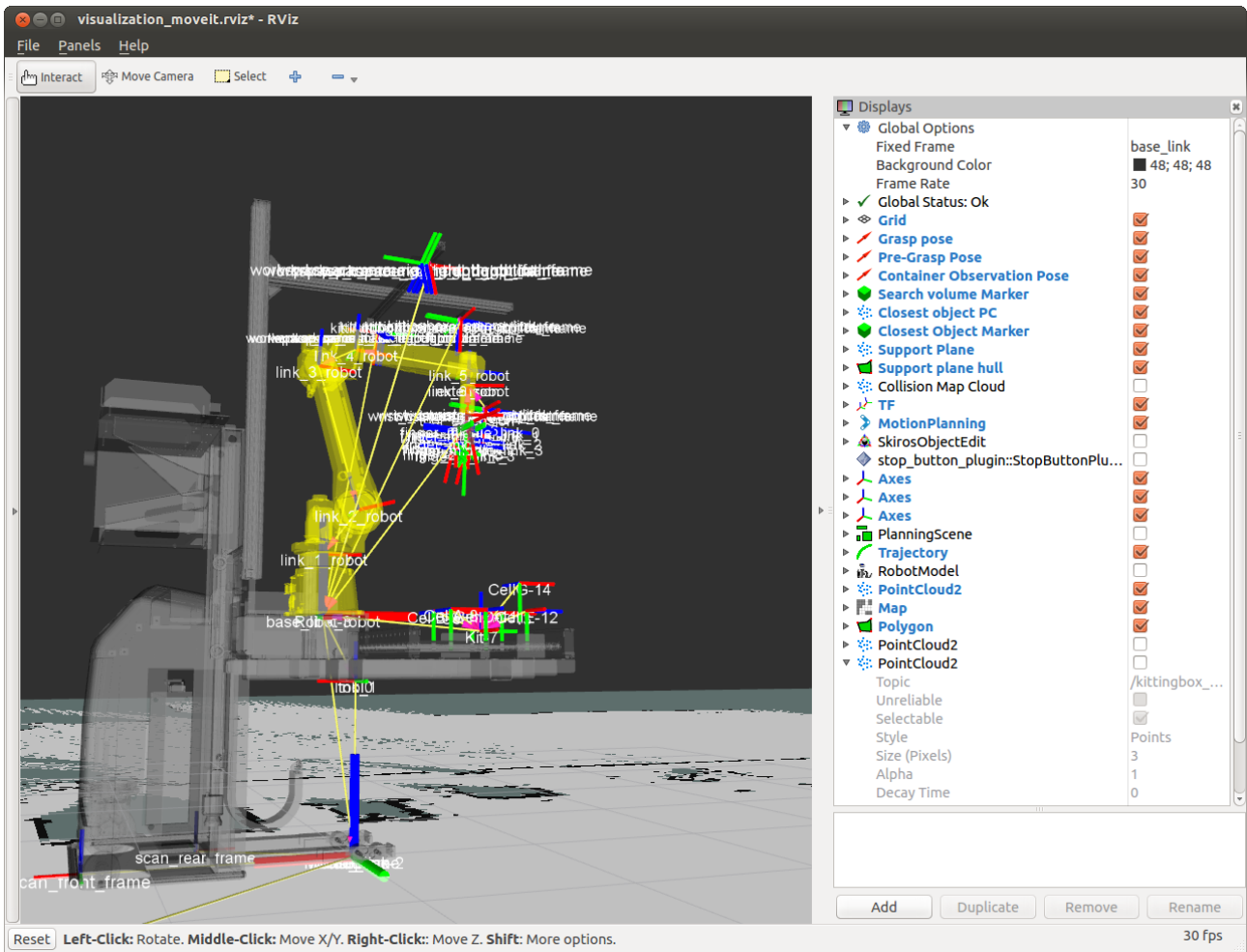


Figure 3 ... and its RViz virtual model for simulation

## 3 Novel System Components

### 3.1.1 Automatic Camera Calibration

AAU has developed an automatic, easier-to-use camera calibration. Most of the openly available camera calibration systems include a checkerboard and the manual movement of the arm to pre-defined calibration points. As already outlined in the previous deliverable, the aim of the novel approach is to calibrate the cameras *without* dedicated markers and in form of an *online* procedure so that the system can remain calibrated all the times. In its present version, the system is calibrated off-line. But since it is so much easier to use, a proper system calibration is not really an obstacle anymore and it can be done on-the-fly if there is reason to believe that strange system behaviour is due to calibration issues.

The recent development of calibration algorithms has been driven into two major directions: (1) an increasing accuracy of mathematical approaches and (2) an increasing flexibility in usage by reducing the dependency on calibration objects. These two trends, however, seem to be contradictory since the overall accuracy is directly related to the accuracy of the pose estimation of the calibration object and therefore demanding large objects, while an increased flexibility leads to smaller objects or noisier estimation methods.

The method we have developed resolves this problem in two steps: First, we derive a simple closed-form solution with a shifted focus towards the equation of translation that only solves for the necessary hand-eye transformation. We show that it is superior in accuracy and robustness compared to traditional approaches. Second, we decrease the dependency on the calibration object to a single 3D-point by using a similar formulation based on the equation of translation which is much less affected by the estimation error of the calibration object's orientation. Moreover, it makes the estimation of the orientation obsolete while taking advantage of the higher accuracy and robustness from the first solution, resulting in a versatile method for continuous hand-eye calibration.

The paper has been submitted to ICRA2017, and can be found in the appendix.

## 4 Storyboards used during the test sprint.

Already for the third test spring we had developed a list of storyboards that were supposed to be tested during the test sprint. The story boards describe realistic situations on the shopfloor, they were identified through a long discussion process during several physical and virtual meetings between all partners.

1. **TEST1: Automation Scenario shows the robot doing its job:** This scenario is the usual scenario in a successful shop-floor application of the robot:
  - a. The robot received a kitting order

- b. starts its round to collect the requested parts: The entire process runs smoothly: The picking placing and the navigation skills work as expected and pre- and post-conditions of the skills catch potential errors. In case of such errors, the planner is able to re-plan.
  - c. At the end the robot delivered its kitting box to a pre-defined location.
2. **TEST2: Start-up in the morning and shutting down after the shift:** The robot is able to connect itself to the logistic planner automatically at start of the shift. At the end of the shift, the robot is driving to its parking location.
  3. **TEST3: Installing a new robot into an existing STAMINA system:** This is ideally working completely automatic.
  4. **TEST4: Starting from scenario 1, test the following cases:**
    - a. Object A is on two different pallets. Pallet 1 is empty. Robot updates the world-model, requests filling of pallet 1, re-plans and uses pallet 2. Next kitting task excludes pallet 1.
    - b. Object A is on two different pallets. Pallet 1 by mistake holds object B. Robot requests attention, re-plans and uses pallet 2.
    - c. Object A is on two different pallets. Pallet 1 by mistake holds object B. Robot re-plans to use pallet 2, but pallet 2 turns out to be empty. Robot calls for human assistance.
    - d. Object A is on three different pallets. Pallet 1 by mistake holds object B. Robot requests attention, re-plans and uses pallet 2. Pallet 2 is empty. Robot re-plans to pallet 3.

When running through the various tests, collect the following relevant data

1. For each mission:
  - a. Number of parts: missions are run with maximally  $n=6$  parts and at least 1 part
  - b. Number of successful missions
  - c. Number of detected failures: These failures are good failures: the system remains in a well-defined state and the robot has called an operator for assistance.
  - d. Number of undetected failures: These failures are bad failures as they leave the state in an undefined state. The aim of the pre- and post-condition checks is to prevent these failures.
  - e. Average execution time per *n-part* mission
2. For each skill:
  - a. Number of attempts
  - b. Number of successes
  - c. Number of detected failures: again, these types of failures are good failure as they allow the task planner to re-plan to continue with task completion.
  - d. Number of undetected failures: again, these are bad failures because they are outside the specs of the system.
  - e. Average execution time of each skill

## 5 Experimental Results

We have run 48 kitting missions with varying number of parts. This resulted in the execution of 615 skills and 1652 skill primitives. The performance and debugging statistics of all missions, skills and skill primitives were automatically recorded on the robot and stored for subsequent evaluation.

All missions were executed using the following parts:

- Starter A
- Starter B
- Alternator A
- Alternator B
- Compressor
- Tube
- Engine support

Starter A and Starter B are both starter engines, but they are physically different parts and are handled as such. Alternator A and Alternator B are physically the same parts, but they were stored in the world model as two distinct parts in order to add additional complexity.

Starters, Alternators and Compressor are well ordered on a pallet and require *depalletizing*.

The engine supports were placed in a Styrofoam fixture in a box to assure that they have a suitable pose for being grasped. The tubes were randomly stored in a box.

All parts had their own location in the kitting area. The boxes with the tubes and the engine support were stored on the lower level of the racks.



Figure 4 shows how tubes are stored in the boxes and how boxes are placed on the racks

### 5.1 Baseline missions

To test the robustness of the system, we ran 27 missions with 1-3 parts as the base-line. Using TEST1 storyboard, 1-3 part missions were requested from the MES. In each mission, the parts were randomly selected from a set of parts consisting of

- Starter A,
- Starter B,
- Alternator A,
- Alternator B,
- Compressor.

The results are summarized Table 1 in row “1-3 parts”. As can be seen, in 24 cases, the robot completed its mission and successfully delivered the complete and correctly assembled kitting box. In 3 cases, the robot had detected problems and aborted the mission. The failure was identified to be due to specific illumination conditions at that particular point in time. The sun was shining directly through the roof windows onto the robot, and the robot was not able to locate the kitting box with its camera system. Once the source of the problem was identified, parameter settings within one skill were adapted and the robot was able to function according to specs. This shows two things: a) unexpected environment conditions can always disturb the system upon startup and initial testing and b) if done right, the software can be adapted to the specific conditions of the test environment. Once solved, the problem had not re-appeared and the STAMINA system was running according to specs through the entire remaining time of the experiments.

## **5.2 Startup in the morning and installing a new robot**

The tests TEST2 and TEST3 were explicitly mentioned in the previous section as they are important requirements from the end-user perspective. The system has been designed to conform with these requirements, and thus TEST2 and TEST3 are merely simple function tests of the stamina system. We were able to show that the stamina robot completes TEST2 and TEST3 without errors.

## **5.3 Complex missions**

Next, more complex missions were tested. Here, we increased the number of parts to 4,5 and 6. We systematically tested the 4 different TEST4 storyboards.

- 4.1. *Object A is on two different pallets. Pallet 1 is empty. Robot updates the world-model, requests filling of pallet 1, re-plans and uses pallet 2. Next kitting task excludes pallet 1.* The necessary updates of the world model via the logistic planner was done within 5 min. The robot was immediately able to execute this mission successfully. Upon detecting the lack of parts on pallet 1, the robot informed the operator (“Alert: Pallet with ID  $\diamond$  empty”), and re-planned to fetch the part from pallet 2. This has been tested twice as a 4-part mission and twice as a 5-part mission.
- 4.2. *Object A is on two different pallets. Pallet 1 by mistake holds object B. Robot requests attention, re-plans and uses pallet 2:* This test was also correctly executed by the robot. Upon detecting a wrong part on Pallet 1, the robot sends a message to the operator (“Alert: Wrong part on pallet 1?”) and continues with re-planning its mission. This has been tested twice as a 4-part mission and twice as a 5-part mission.
- 4.3. *Object A is on two different pallets. Pallet 1 by mistake holds object B. Robot re-plans to use pallet 2, but pallet 2 turns out to be empty. Robot calls for human assistance:* This test case is similar to the previous one with the difference that the robot is in this case not able to complete its mission: The robots first detects the wrong part and sends an alert to the operator. Then, it re-plans to pick from pallet 2 which it finds empty. After sending again an alert to the operator

about Pallet 2 being empty it attempts to re-plan. The re-planning fails because there is no third pallet that could provide the required part. The robot therefore aborts the mission with a notification to the operator (“Alert: No plan to complete the mission”). It is interesting to mention that all the task planner on the robot can do is to attempt to find a plan. If a plan is available, it is usually found within  $< 1s$ . We have therefore set a 2s-threshold so that the planner reports “no plan found” and aborts the planning in case no plan could be found within 2 seconds. The reason is because a planner is not able to assess *why* a plan was not found, i.e., all it can do is to give a binary answer *plan/no-plan*. This has been tested once as a 4-part and once as a 5-part mission.

4.4. *Object A is on three different pallets. Pallet 1 by mistake holds object B. Robot requests attention, re-plans and uses pallet 2. Pallet 2 is empty. Robot re-plans to pallet 3:* This last test is a mixture of all the previous ones, and it demonstrates the power of using a task planner and the skill-based approach. Using skills and a task planner, the robot is able to dynamically find a solution automatically that is within the space of possible skill sequences. Alternatively, the more classic approach of not using a planner would require to analyze the scenario to identify all possible situation that might appear on the shop-floor. This is a highly time-consuming task which, in addition, leads to a robotic system being scenario specific.

All of the above story boards have been tested several times with 4 and 5-part missions. Table 1 summarized the results. We have done 7 4-part missions and 9 5-part missions. Tests 4.1, 4.2 have been run twice for both 4 and 5-part missions, test 4.3 has been tested once for a 4-part and once for a 5-part mission Test 4.3 was defined to fail with a *detected (good) fail*. In Table 1, these two detected (good) fails for the 4-part and 5-part mission can be seen in column “Detected (good) fails”.

In addition to the TEST4 storyboards we have also tested TEST1 with 5 and 6-part missions. Here it became apparent that the picking works reliably only for de-palletizing tasks, but not for bin-picking tasks. Picking the tube requires bin-picking, picking the engine support was similar to de-palletizing, but since the engine supports were placed in small boxes, the picking was much more challenging than picking from pallets due to the confined space. Tube and engine support picking resulted into 3 detected errors for three 6-part missions.

It is important to point out that, contrary to the previous test sprint, we did not observe any undetected fails. In other words, the system was always within a well-defined system state. This was achieved by a) appropriate debugging of the errors discovered in the previous test sprint and b) by adding one additional but apparently important sensing mechanism: *binary test for object in the gripper*. It was shown in the previous test sprint that the system often entered an undefined state because it was not able to detect that a picked object had fallen out of the gripper.

In column “Execution Time” of Table 1 it can be seen that the 6-part missions were executed faster than the 5-part missions, while the execution time of the other missions was roughly linear with number of parts. 6-part missions were tested towards the end of the experiments. By that time, we had already seen that the skills are running very reliable and we were able to tune the skill parameters towards faster execution without loss of performance.

The success rate in Table 1 was calculated based on the ratio between successful completion of missions verses the fails, as this is how the enduser would evaluate it. From a scientific perspective, even the missions that failed with a detected failure should be considered to be successful because the missions were executed according to specs, and the mission failure was due to *faults in the kitting zone*: wrong and missing parts. In other words, **the kitting zone was out of spec, and not the robot.**

Mission type	Kitting Tasks	Success according to STN	Detected (good) fails	Undetected (bad) fails	Execution Time	Success Rate
6 parts	5	2	3	0	880,36s	40%
5 parts	9	8	1	0	916,01s	88%
4 parts	7	6	1	0	723,77s	86%
1-3 parts	27	24	3	0	634,22s	89%

Table 1 summarizes the results of the test missions

## 5.4 Testing the Robot skills

All missions from the previous section were executed by using three different kinds of skills: The navigation skill (`drive_freiburg`), the placing skill (`place_aau`) and the picking skill (`pick_bonn_aau`). All skills were extensively tested, either through the missions or explicitly by manually testing them. The performance statistics of the skills were automatically recorded and evaluated. Table 2 provides a summary of these statistics. Column “Attempts” in Table 2 shows the overall amount of executions per skill. The amount of attempts is larger for skills that failed more often. The amount of success should be roughly the same for all skills. One can see that we have tested the pick skill and the drive skill manually more often than the place skill.

When reading Table 2 it is important to keep in mind that the skills are executed based on a task planner, planning and re-planning the missions. This means, if a skill fails, the system re-plans which leads in most cases to a re-attempting of the skill, either immediately or at a later time. This means that eventually, all skills were executed successfully, even if a detected (good) failure of a skill resulted into re-planning. This also means that the column “Success” in Table 2 reflects how often each skill was successfully executed **on the first try**. Eventually, as can be seen by the mission experiments in the previous section, all skills were executed successfully. Only in three cases, a skill truly failed (as a good fail): the `place_aau` skill in the baseline mission experiments (Sect 5.1). As discussed there, the problem was solved with a parameter update of skill to allow it to cope with the illuminations conditions.

Skill	Attempts	Success	Fails	Undetected fails	Average time	Success
drive_freiburg	222	181	41	0	18,30	82%
place_aau	156	141	15	0	32.06	90%
pick_bonn_aau	237	154	82	0	98.68	65%

Table 2 summarizes the test results of the skills

This insight results into the following observation. Given the success rate for each skills in Table 2 as *probabilities of success*, the theoretical probability of executing a  $n$ -part mission is

$$(\mathcal{P}(\text{drive}) \cdot \mathcal{P}(\text{pick}) \cdot \mathcal{P}(\text{place}))^n \cdot \mathcal{P}(\text{drive}).$$

The probabilities are explicitly calculated in the rightmost column in Table 3. These are the probabilities that apply in the case that we *hardwire* a robotics program in the classical way as a fixed sequence of macros. The impressive difference between the performance the hardwired system would theoretically have on average and the de-factor performance is due to the pre- and post-condition checks of the skills and the planner being able to handle unexpected events.

Mission type	Success Rate	Theoretical success rate
6 parts	40%	1%
5 parts	88%	3%
4 parts	86%	5%
3 parts	89%	11%

Table 3 shows the de-factor success rates and the theoretical success rates

## 5.5 Testing skill primitives

We have tested altogether 1652 skill primitives. The success rate of “locate\_bonn” is the same as last time. Skill primitive “arm\_motion” has been improved over last time and is now more likely to find a suitable arm movement for object picking and placing. The rate of the “kittingbox\_registration” is slightly lower due to the illumination problems we had mentioned above. While in the previous test sprint the success rate was 100%, we have this time only 94% due to 9 failures. Most of the failures have probably happened before the update of the parameters, but even after the parameter update there might have possibly been some failures, but they were compensated through replanning.

The “ff\_planner” has a success rate of 91% due to 21 fails. An ff\_planner-fail is counted when no plan could be found. In that sense, the planner itself did not fail, just just failed to provide a plan. Not finding a plan was in all our experiments due to the fact that there was none. E.g. in TEST4.3, there was no plan possible and the robot had to call for help.

The novel skill primitive “gripper\_oc” was used to detect if an object was present in the gripper. This skill-primitive provided an additional sensing capability, e.g., the ability to check if a pick-skill was successful, if the precondition-check of a place skill, object-in-gripper?, is true.

Primitive	Attempts	Success	Fails	Undetected fails	Average time	Success
locate_bonn	204	198	6	0	3,36	97%
arm_motion	650	639	11	0	11.41	98%
kittingbox_registration	237	223	9	0	5,64	94%
ff_planner	252	229	21	2	6,39	91%
gripper_oc	309	309	0	0	2,17	100%

## 6 Conclusions

In terms of our own ambitions this test sprint was also a great success: We have truly reached the goal of the project.

1. The complete system is now really running robustly. We have done a large number and a big variety of experiments to verify and validate the system.
2. We have tested a large variety of challenges and test story board, and the robot was able to handle those completely according to spec.
3. We have been able to completely remove undetected failures, i.e., the failures that the robot is not able to detect. It is very important to identify which errors matter and what sensory devices are needed.
4. The only short-coming is the limited picking capability, i.e., we can only do de-palletizing. However, it turns out that that this is what PSA is mostly interested in. PSA plans to use the STAMINA system to handle heavy parts such as compressors and alternators, and these come on pallets.

The system as shown another issue: that the use of skills and a planner can be very powerful compared to robotic systems that are programmed in a classic way. The planner allows to decouple the scenario from the system. The scenario information is completely encoded in the world model.

That means an adaption of the system to a new scenario is easily possible. In classically programmed robots, the scenario knowledge is encoded also in the actual program code, and this makes the adaption of a system to a new scenario difficult. Consider, e.g., how one would deal with the TEST4 scenarios: all possible reactions of the robot would have to be hardcoded in the program, e.g., through a switch-construct. To hardcode the behaviours would require to analyze the scenario

to be sure that the robot can handle all possible situations. What one builds here into the programming code of the system is really a *model* of the scenario. To adapt such a program to a new scenario would require to analyze the new scenario and to correct the program code. Our approach, on the other hand, the robotic system itself is *model-free*, because no such analysis is necessary. The pre- and post-conditions are not scenario dependent because they only depend on the skill they belong to and the code the skill encapsulates.

## 7 Appendix

The appendix contains recent relevant publications:

Bjarne Großmann and Volker Krüger. Continuous hand-eye calibration using single points. Submitted for review to ICRA 2017.