DANISH TECHNOLOGICAL INSTITUTE

Final Demo Report

AAWSBE1

Title:
AAWSBE1

Prepared for:
ECHORD++ Call 2

Prepared by:
Danish Technological Institute
Forskerparken Fyn
Forskerparken 10
5230 Odense M
Robot Technology

April 2018
Author: Carsten Panck / Michael Nielsen /Rasmus Johansen

Table of Contents

1. Introduction

The robot task is to pick a number of objects from a vision frame. A vision frame is a snapshot of e.g. 600x600mm of the conveyor plane. A frame is triggered every 400mm and tracked with conveyor tracking (encoder) by both the robot and vision system. The frame is calibrated metrically in the plane of the conveyor in vision and robot. The distance between vision and robot workspace can be multiple of such frames. It is important to refer to each of such frame with a unique ID. The robot gives this to the vision system:

- Robot sends a trigger every X mm of conveyor travel
- Vision system receives trigger connect to robot and receives FrameID
- Vision system produces picks
- DTIPickWare module sorts and optimizes pick order
- Pickware sends the picks to robot
- Robot detects false vacuum and false weight and sends a list of failed picks

2. DTIPickWare

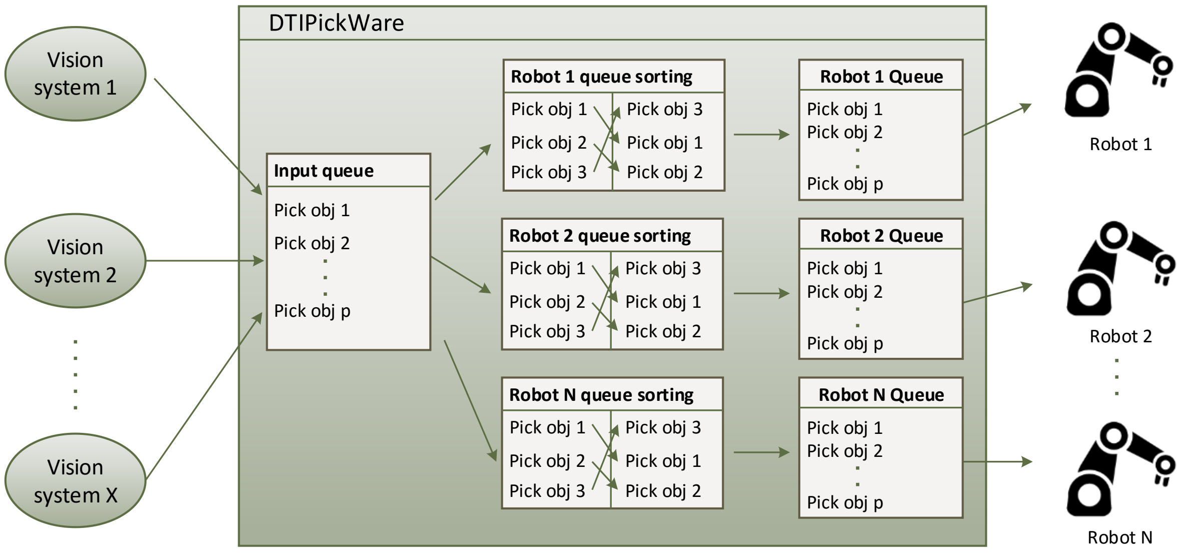
DTIPickWare bridges the gap between the vision system and the robot(s) employed to perform the picking and handling tasks as defined by the integrator.

Key features of the module are:

- Flexible prioritization of candidate pick objects sent to DTIPickWare
- Multiple concurrent client connections ie. Vision systems delivering pick objects
- Connection to multiple robots facilitating large throughput and diverse geometries
- Queue handling to each robot in order to even out object buffer capacity
- Generic client/server interfaces using TCP/IP connections

2.1. System overview

Below is a block diagram of the basic functionality within DTIPickWare:



When new pick objects added to DTIPickWare, they undergo sorting, based on predefined prioritization algorithm. When all unqueued objects have been sorted, they are sent to the designated robots for picking. If the queue is too large for the buffer in the robot, the remaining pick objects are left in the queue for transmission when the robot regains capacity to receive new pick objects.

Pick objects store all data related to a single object that needs to be picked, which is detailed in the section below.

2.2. Pick object

Each pick object from the vision system encapsulates the data below:

Data	Comment
FrameID	The specific ID that relates to the image and position on the conveyor. This ID is initially sent from the robot to the vision system, and the ID is relayed to DTIPickWare.
X	X Value in mm of the object on the conveyor
Y	Y Value in mm of the object on the conveyor
Z	Z Value in mm of the object on the conveyor
ClassID	The class (integer) that the object belongs to. This could be Mobile Phone, plastic, metal, battery etc.
ItemID	Unique ID of the object that is given by the vision system
ToolID	Corresponds to the robot ID that is best capable of picking the actual object
DstID	ID of the sorting bin that the object should place the object in
Value	User defined value that is derived from a prioritization algorithm
WeightMin	Minimum estimated weight in grams of the item
WeightMax	Maximum estimated weight in grams of the item

A pick object is generated for each detected item in the image acquired by the Vision system. The pick object is populated with position values read from the values received by the vision system.

Based on the size and type of the products detected, the Minimum and maximum weight limits are estimated and applied to the pick object before transferring to DTIPickWare.

The collection of pick objects from the current image is sent to DTIPickWare for sorting and queueing for the robots.

3. TCP Server for client connections from vision systems

The DTIPickWare module can either be integrated into the vision solution, where Pick objects can be sent directly through the program, or can exist separately from the vision systems used for the specific task. DTIPickWare facilitates a TCP server to make this happen. Client vision systems connect to the server and send collections of pick objects as they are being detected.
The objects are sorted and prioritized before passing them on to the robot.
The protocol for sending Pick objects over TCP connection is the following:

```
FrameID, num_objects;
ItemID, ClassID, value, X, Y, Z, WeightMin, WeightMax, ToolID, DstID;
ItemID, ClassID, value, X, Y, Z, WeightMin, WeightMax, ToolID, DstID;
```

Initially in the string, the frame ID and the number of objects are sent. Subsequently, all objects sent, separated by semi colons. Each item in the object is delimited by a comma.

4. TCP Client connections to robot TCP servers

Each robot connected to the system, acts as a TCP server that handles a client connection from the DTIPickWare module.

When pick objects have been added to DTIPickWare, they are passed to the prioritization and sorting entity belonging to the specific robot. The robot is determined from the *ToolID* in the pick object.

The protocol for sending pick objects to robot TCP servers is the following:

1. FrameID, and the number of objects is sent to the robot
2. All objects are sent one after each other, as 4-byte values in this order, making a total of 32 byte:
    a. ItemID
    b. ClassID
    c. X
    d. Y
    e. Z
    f. ToolID
    g. DstID
    h. WeightMin
    i. WeightMax
3. If all values are successfully received, the acknowledge string: 0,5 is returned.

Possible robot acknowledge return values for pick objects

When a robot has processed a pick object by picking it (or trying to), it acknowledges DTIPickWare of the result, which can be any of:

- 1: OK – Object Picked successfully
- 2: ERROR – Vacuuum not obtained when picking object
- 3: ERROR – Object dropped during movement
- 4: ERROR – Weight wrong

Based on the results, the object is either resent to the robot or purged from the queue. The behavior can be decided from one implementation to the other.

5. Object Prioritization

Prioritization methods

Depending on the desired functionality of the system, a number of different methods exist to prioritize the incoming pick objects. Some examples of prioritization methods are listed below:

Method	**Comment**
By object value	The higher the value of the item, the higher priority
By X position	This is the standard used in most picking systems, sorting all objects based on their position along the conveyor. In many situations this is not optimal, and could yield fewer picks than if other considerations were taken into account.
By Z position	

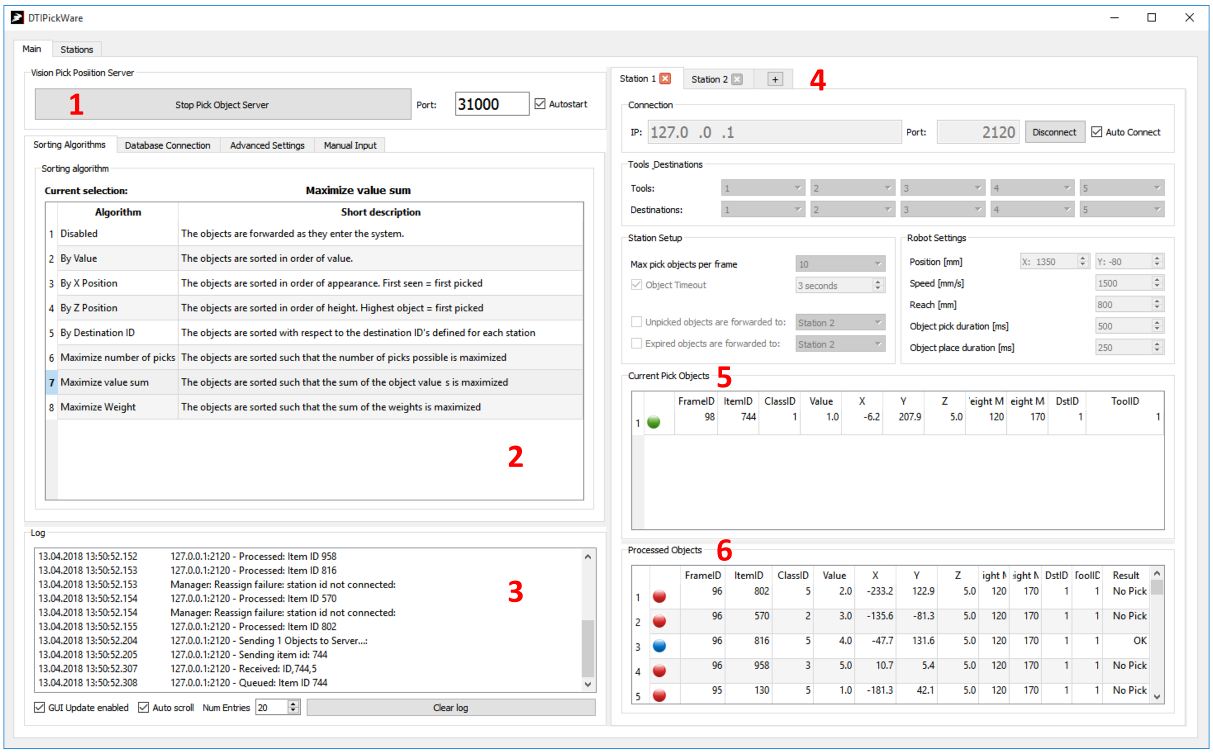
Maximize number of picks	Prioritize based on position on conveyor belt and distance to the robot.
Maximize value sum	Design the pick order such that the robot always picks the objects that collectively yield the largest sum of values of the objects.
By destination ID	Prioritize one destination to others.
Maximize weight	

6. Graphical user interface

The following section describes the graphical user interface of the DTIPickWare.

6.1. Main Window

The main window below is presented when starting the program:



Port: 31000 ☑ Autostart

Current selection: **Maximize value sum**

	Algorithm	Short description
1	Disabled	The objects are forwarded as they enter the system.
2	By Value	The objects are sorted in order of value.
3	By X Position	The objects are sorted in order of appearance. First seen = first picked
4	By Z Position	The objects are sorted in order of height. Highest object = first picked
5	By Destination ID	The objects are sorted with respect to the destination ID's defined for each station
6	Maximize number of picks	The objects are sorted such that the number of picks possible is maximized
7	Maximize value sum	The objects are sorted such that the sum of the object value s is maximized
8	Maximize Weight	The objects are sorted such that the sum of the weights is maximized

```
13.04.2018 13:50:52.152     127.0.0.1:2120 - Processed: Item ID 958
13.04.2018 13:50:52.153     127.0.0.1:2120 - Processed: Item ID 816
13.04.2018 13:50:52.153     Manager: Reassign failure: station id not connected:
13.04.2018 13:50:52.154     127.0.0.1:2120 - Processed: Item ID 570
13.04.2018 13:50:52.154     Manager: Reassign failure: station id not connected:
13.04.2018 13:50:52.155     127.0.0.1:2120 - Processed: Item ID 802
13.04.2018 13:50:52.204     127.0.0.1:2120 - Sending 1 Objects to Server...:
13.04.2018 13:50:52.205     127.0.0.1:2120 - Sending item id: 744
13.04.2018 13:50:52.307     127.0.0.1:2120 - Received: ID,744,5
13.04.2018 13:50:52.308     127.0.0.1:2120 - Queued: Item ID 744
```

☑ GUI Update enabled ☑ Auto scroll Num Entries 20

IP: 127.0 .0 .1 Port: 2120 ☑ Auto Connect

Tools:	1	2	3	4	5
Destinations:	1	2	3	4	5

Max pick objects per frame: 10
☑ Object Timeout: 3 seconds
☐ Unpicked objects are forwarded to: Station 2
☐ Expired objects are forwarded to: Station 2

Position [mm]: X: 1350 Y: -80
Speed [mm/s]: 1500
Reach [mm]: 800
Object pick duration [ms]: 500
Object place duration [ms]: 250

		FrameID	ItemID	ClassID	Value	X	Y	Z	'eight M	eight M	DstID	ToolID
1	🟢	98	744	1	1.0	-6.2	207.9	5.0	120	170	1	1

		FrameID	ItemID	ClassID	Value	X	Y	Z	ight N	ight N	DstID	ToolID	Result
1	🔴	96	802	5	2.0	-233.2	122.9	5.0	120	170	1	1	No Pick
2	🔴	96	570	2	3.0	-135.6	-81.3	5.0	120	170	1	1	No Pick
3	🔵	96	816	5	4.0	-47.7	131.6	5.0	120	170	1	1	OK
4	🔴	96	958	3	5.0	10.7	5.4	5.0	120	170	1	1	No Pick
5	🔴	95	130	5	1.0	-181.3	42.1	5.0	120	170	1	1	No Pick

Each element is described below:

- **1:** *Vision Pick Position Server*
  - The server for incoming Vision connections. These connections are for delivering the object positions for each frame acquired by the camera. Multiple vision stations can be connected to DTIPickWare simultaneously.
- **2:** *DTIPickWare Setup Tabs*
  - These tab pages define the behavior of DTIPickWare. Each page is explained in detail below.
- **3:** *Log Window*

  - Outputting relevant information relating to connections, pick positions and errors.
- **4:** *Pick Stations*
  - Each robot connected to DTIPickWare is defined as a *Station*. Here it is possible to set the IP address of the Robot along with parameters that define the behavior:
    - Tools and Destinations
      - IDs of the tools attached to the robot
      - IDs of the destinations that the robot can drop objects into
    - Station Setup
      - Maximum number of objects per frame that the system can take
      - Timeout (seconds) definition.
      - Forwarding of unpicked objects to specific station
      - Forwarding of expired objects to specific station
    - Robot Settings
      - Position of the robot in global coordinate system (X,Y)
      - Speed of the robot in mm/s
      - The reach of the robot arm in mm
      - Time for the robot to pick an object in ms
      - Time for the robot to drop an object in ms
  - The '+' symbol next to the stations tab is used to add more stations to the system. A single DTIPickWare instance can in theory service an unlimited number of robots.

- **5:** *Current Pick Objects*
  - This list shows the current objects that have been sent to the station from DTIPickWare. The color of the icon on the left reflects the current state of the object:
    - Yellow: The position has been sent, but not yet acknowledged by the station
    - Green: The position has been acknowledged and is queued for picking.

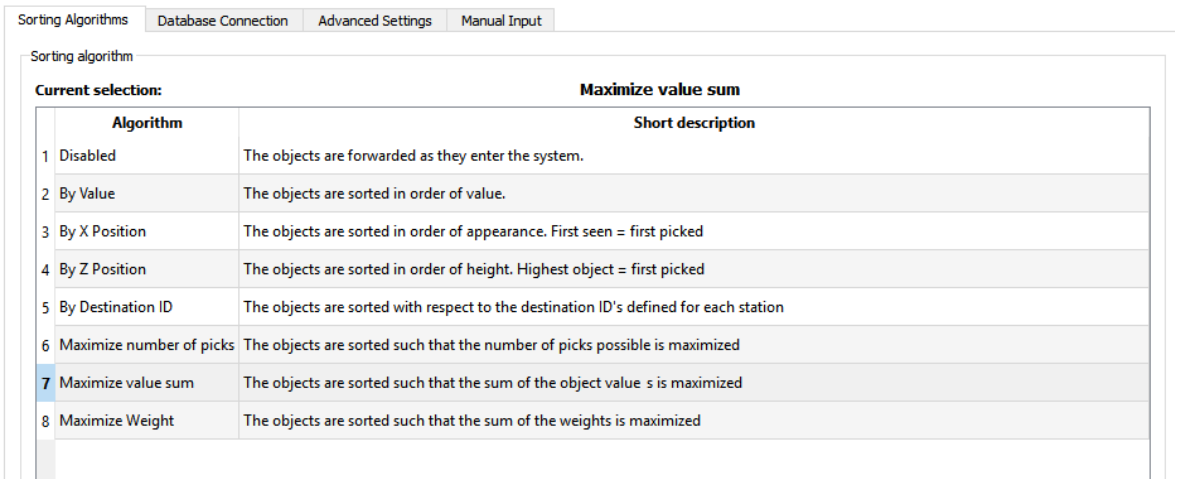
- **6:** *Processed Objects*
  - This list shows pick objects that have been processed by the robot or that timed out. Also here, the color coding of the icon defines the states of the objects:
    - Blue: Object picked successfully
    - Red: Failure in picking object, the result column shows which error is returned, that could be one of the following:
      - Vacuum not obtained when picking object
      - Object dropped during movement
      - Weight wrong
    - Magenta: The object expired (timed out). If enabled, the object is forwarded to the station selected to retry picking.

6.2. Setup tabs

The setup tabs are used to define the general behavior of DTIPickWare. Each page is explained in the sections below.

6.2.1. Sorting Algorithms

On the sorting algorithms page, the user can select which algorithm to prioritize objects after.



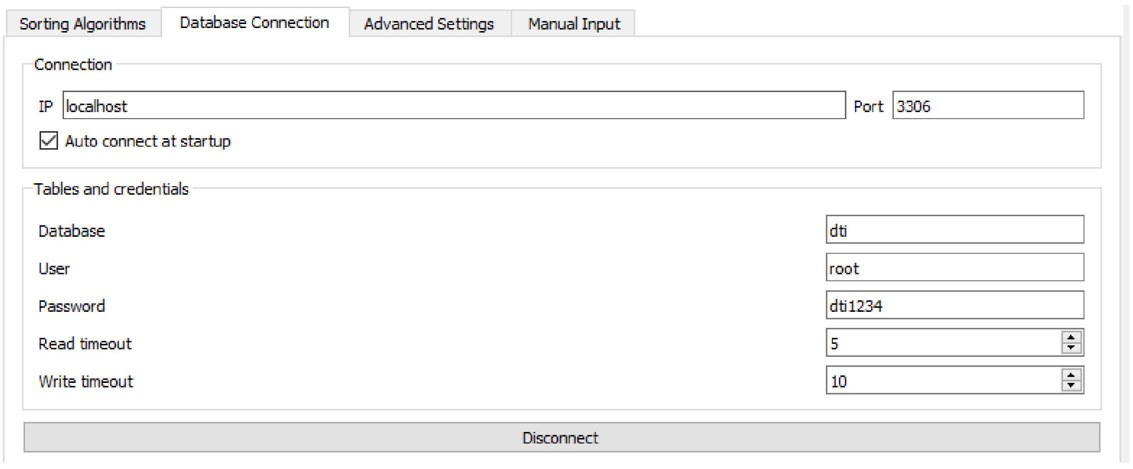
Current selection: **Maximize value sum**

	Algorithm	Short description
1	Disabled	The objects are forwarded as they enter the system.
2	By Value	The objects are sorted in order of value.
3	By X Position	The objects are sorted in order of appearance. First seen = first picked
4	By Z Position	The objects are sorted in order of height. Highest object = first picked
5	By Destination ID	The objects are sorted with respect to the destination ID's defined for each station
6	Maximize number of picks	The objects are sorted such that the number of picks possible is maximized
7	Maximize value sum	The objects are sorted such that the sum of the object value s is maximized
8	Maximize Weight	The objects are sorted such that the sum of the weights is maximized

Each algorithm has a short description next to it. A more detailed description of each prioritization algorithm is given in the section about Prioritizing objects on page 12.

6.2.2. Database Connection

The Database Connection page makes it possible for the user to connect DTIPickWare to a SQL server to store the processed pick objects, in order to perform statistics on the performance of the system.



Connection

IP localhost Port 3306

☑ Auto connect at startup

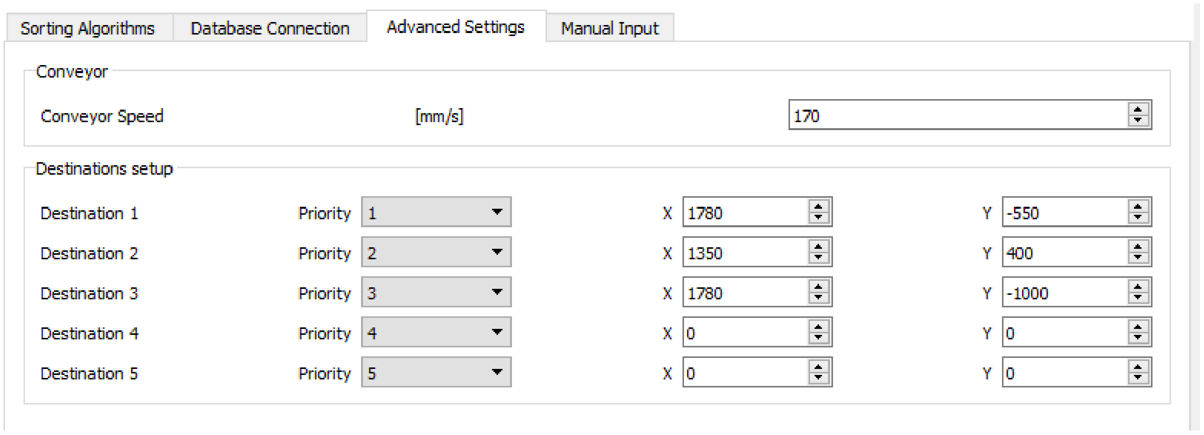
Tables and credentials

Database	dti
User	root
Password	dti1234
Read timeout	5
Write timeout	10

The fields above are all required to make a successful connection to the SQL server. A designated table is also required in order to insert the records properly.

6.2.3. Advanced Settings

The Advanced Settings page is used to setup the physical environment. This is important in order for the advanced sorting algorithms to function properly. For some of the algorithms, the system needs to know various distances in order to create the cost function calculating the times needed to pick and drop the objects delivered by the vision system. This is described in detail in the section called Prioritizing objects on page 12.


Sorting Algorithms | Database Connection | Advanced Settings | Manual Input

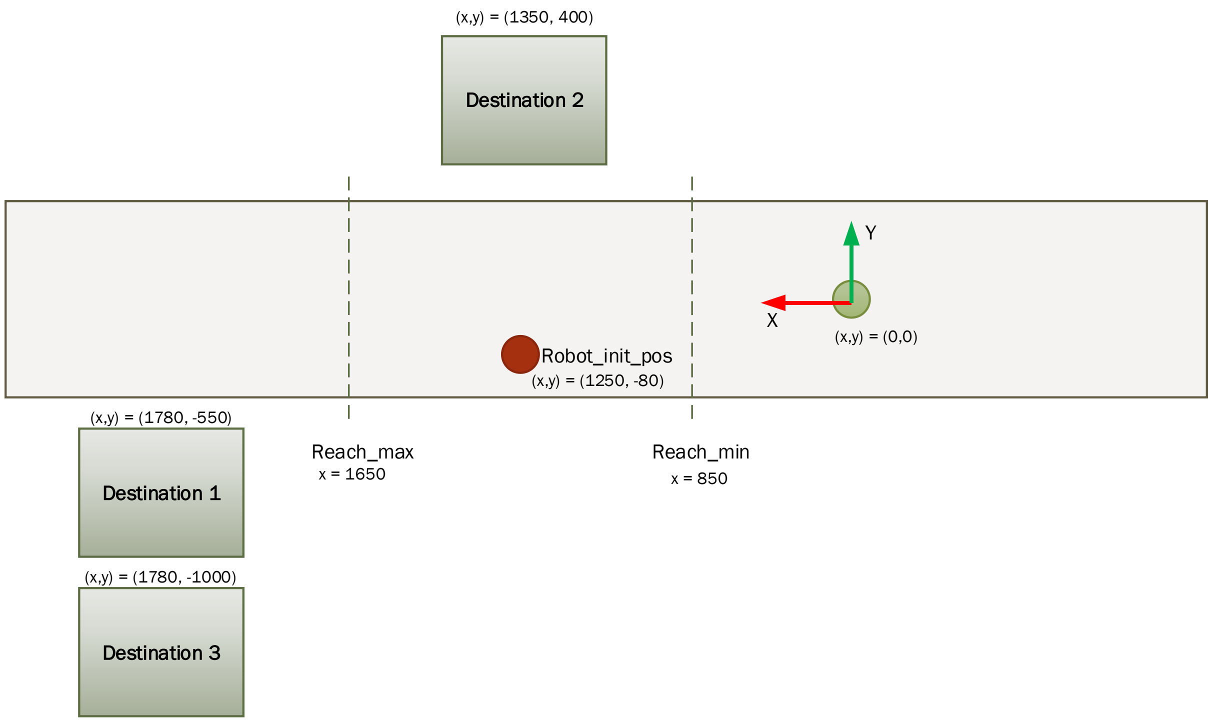
Conveyor

Conveyor Speed	[mm/s]	170

Destinations setup

Destination 1	Priority 1	X 1780	Y -550
Destination 2	Priority 2	X 1350	Y 400
Destination 3	Priority 3	X 1780	Y -1000
Destination 4	Priority 4	X 0	Y 0
Destination 5	Priority 5	X 0	Y 0

What is needed for the system to have knowledge about, is the speed of the conveyor, as well as the position and priority of the destination bins placed around the robot. The origin of the coordinate system is positioned in the center of the camera, and all measurements relates to this. The figure below shows the current layout as defined in the above example:

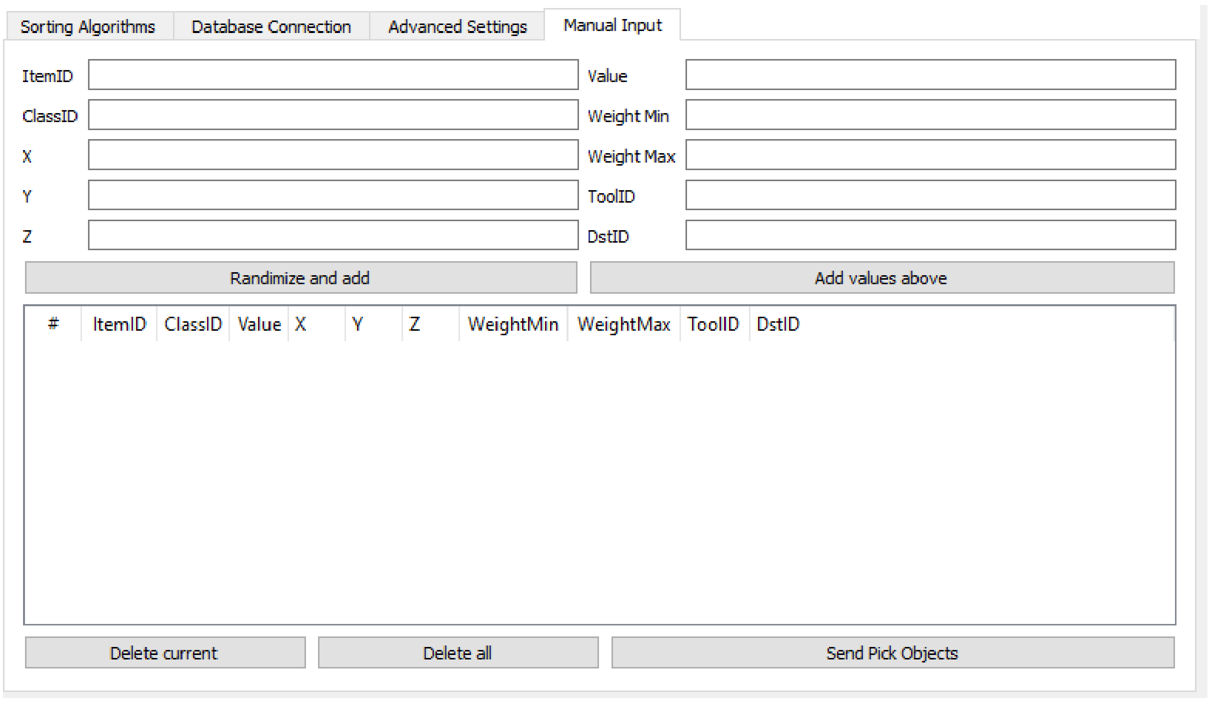


The initial position of the robot is not defined on the advanced page, but on the station setup page described earlier.

The figure also shows the minimum and maximum reach for the robot. An approximation of the circular reach is given as a rectangular area where the robot can grasp objects.

6.2.4. Manual Input

The Manual Input page is made solely for debugging purposes and should not be used when the program is running online.



The page is used to simulate pick objects and to send them for sorting and assigning to robots.

7. Prioritizing objects

One of the key features of DTIPickWare is, that custom prioritization methods can be employed to obtain the functionality one wishes for the specific sorting system wanted.

The algorithms used for the desired functionalities vary in complexity, and are described in the following sections.

The priority of an object is directly associated to the order of which the object is being picked in the robot queue.

The methods of prioritizing objects are described in the following sections.

7.1. Simple algorithms

The simple sorting algorithms are purely based on the values of the pick objects themselves and relations to other objects in the same frame are thus not taken into consideration.

- ***Value***
  - Prioritization is based on the value of the object. The higher the value, the higher the priority

- ***By X position***
  - The objects are sorted based on their position along the conveyor. The object that first enters the picking area of the robot, gets the highest priority.

- ***By Z position***
  - Sometimes it is desirable to pick the highest objects first, in order to protect the robot tool and to increase the rate of successful picks. Therefore, the objects are sorted such that the highest objects get the highest priority.

- ***By Destination ID***
  - Objects of a certain type that needs to go into specific bins (destinations) may be of higher importance that other types. Therefore, prioritizing based on the priority of the destinations can be used to obtain this functionality. On the advanced settings page, the priorities of the different destinations are defined. The lower the value, the higher the priority. Ie. a destination with a priority value of 1 is more preferable compared to a destination with priority value 5.

7.2. Advanced algorithms

Based on a cost function maximizing the sum of the priorities of the picked objects, while minimizing the distance the robot should travel in order to pick as many objects as possible.

Distances from robot to each object must be calculated and the time derived between picks minimized.

Then each object in a vision frame will be lines up on a graph and the time it takes to pick the list in that order till be computed, to see how many picks it will reach. The sum of value will then be computed.

For example, look at 3 object in a frame (each – is a movement * is a box):

- 1 –*- 2 –*- 3 -*-

- 1 -*- 3 -*- 2 -*-

- 2 -*- 1 -*- 3 -*-

- 2 -*- 3 -*- 1-*-

- 3 -*- 1 -*- 2 -*-

- 3 -*- 2 -*- 1 -*-

Each graph must then be computed how deep into the graph the robot has time for. Maybe items 2 -3 -1 pick order is possible, while 1 – 3 has more value, even if there is no time to pick 2.

For all combinations of object orders (permutations), the accumulated time will be calculated for each pick and drop. When this has been calculated, depending on the chosen sorting algorithm, all possibilities will be tested that maximizes output while performing fastest. The different advanced algorithms are listed below:

- ***Maximize number of picks***
  - All permutations are sorted such that the most picks are possible at the shortest time. Initially the list is sorted based on number of possible picks within the time given for the robot. After this, if there are multiple solutions that yield the same number of picks, the one that completes in the shortest time, is chosen.

- ***Maximize value sum***
  - All permutations are sorted such that the sum of the values of the objects are maximized within the given time frame for the robot. That means, in a scenario of 3 objects, it may choose to only pick two objects because they are of high value.

- ***Maximize weight***
  - This algorithm maximizes the total weight of the picked objects, and prioritizes the objects after this principle. Those objects that cannot be picked within this timeframe will be discarded.

Custom algorithms can be added to the system without needing to change layout of the user interface, which makes upgrading algorithms a relatively simple task.

8. Robot Program

ABB IRB1200 was used as test platform. The robot program consisted of two tasks:

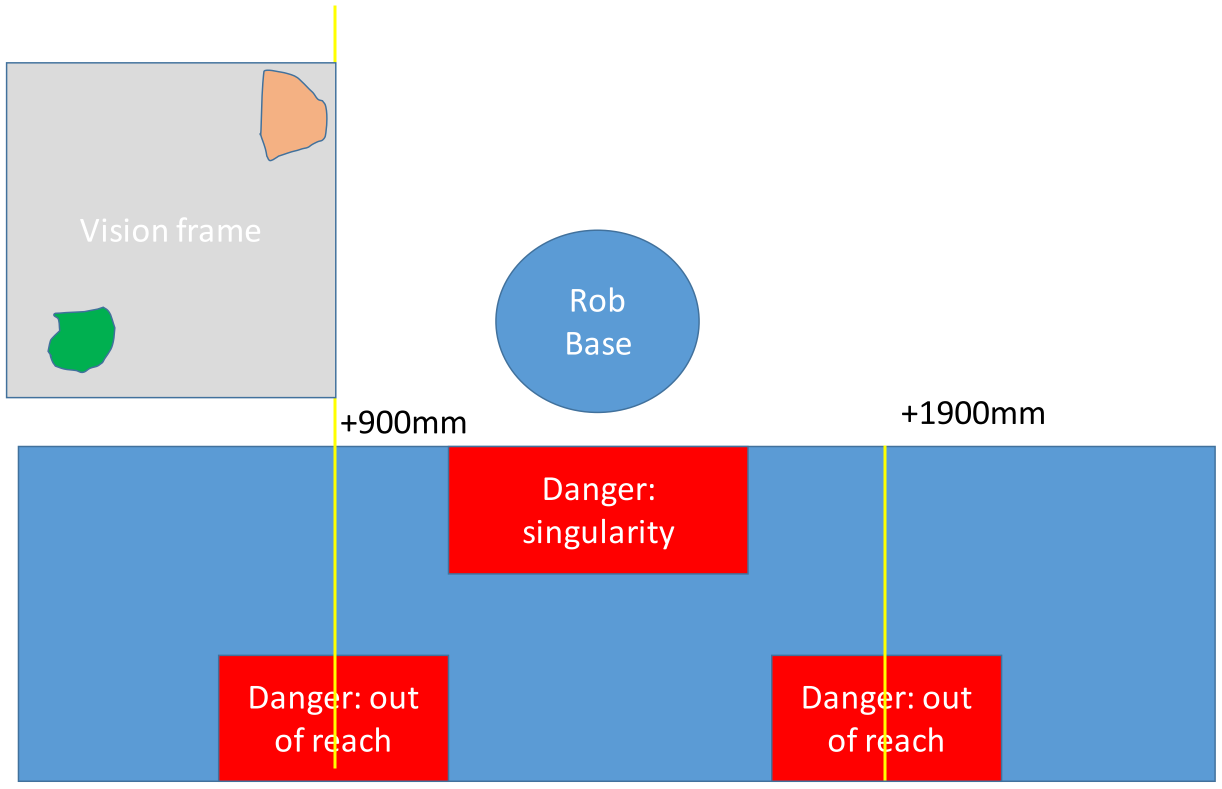
- Pickware_Com
- Pickware_Rob

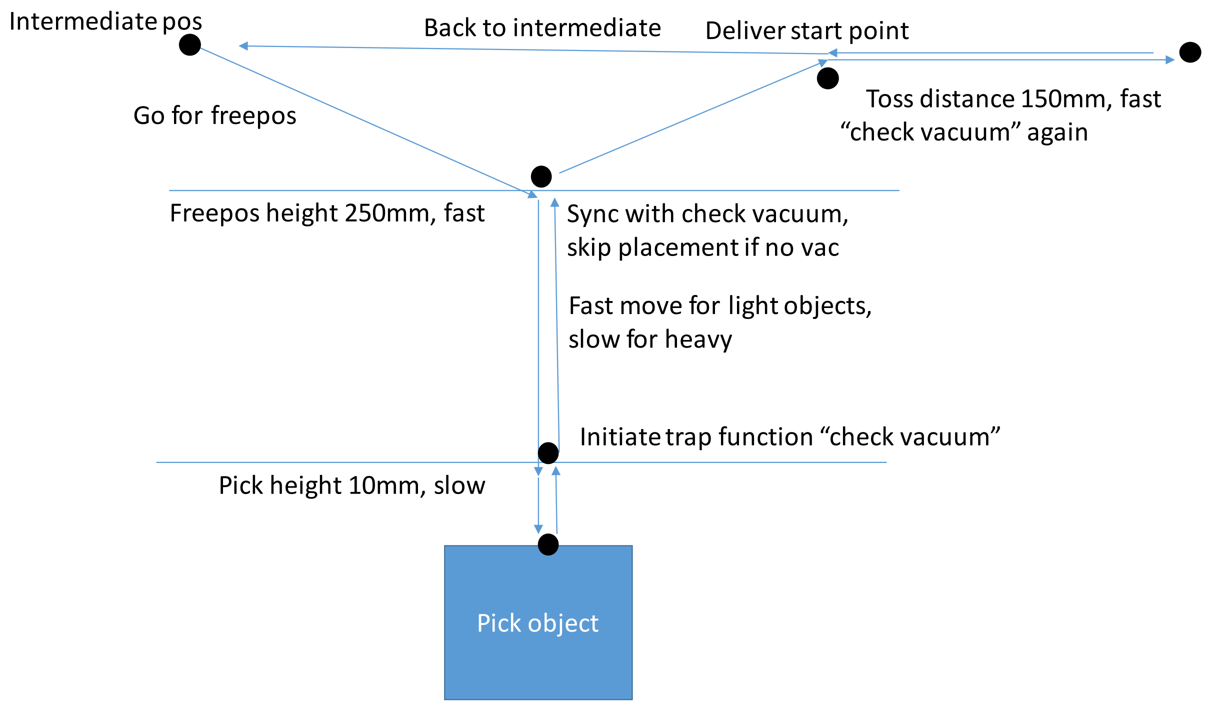
The com task handled the two connections – one for pickware, receiving picking jobs, and one for the vision algorithm, for giving frame ids.

The Rob task handled the pick and place routine:

- Wait for conveyor frame to enter the robot area (e.g. +900mm from vision).
  - This means that if the frame is 400mm long. Object can be placed 700-1100mm from the center of the camera. Ergo, they also vary in distance from the robot itself.
  - So we can choose to delay the frame until the entire frame is reachable, but our sticker test showed that the time the frame is pickable by the robot is too short this way.
- Start looking at the pick list for the frame.
  - If the object is in reach (regarding the OOR and the singularity zone), pick it, otherwise wait for it to pass the danger zone to the left and at the base.
  - The limits are different for gripper 1 and 2.
- Initiate the pick pattern shown in the figure, go back to intermediate position

- Freepos height is adjustable in the teach pendant app
- Sync timers when to start and stop vacuum/blow , check vacuum is also adjustable on in the app.




Intermediate pos
Back to intermediate
Deliver start point
Go for freepos
Toss distance 150mm, fast
"check vacuum" again
Freepos height 250mm, fast
Sync with check vacuum, skip placement if no vac
Fast move for light objects, slow for heavy
Initiate trap function "check vacuum"
Pick height 10mm, slow
Pick object

9. Pick Performance

The pick performance was first tested using post it notes because they loop around and we could place them in clusters to stress the system, making sure it always had something to pick, close and far from the base. This way we could also test the longevity of the system over many hours, day after day, without having to feed it waste constantly.

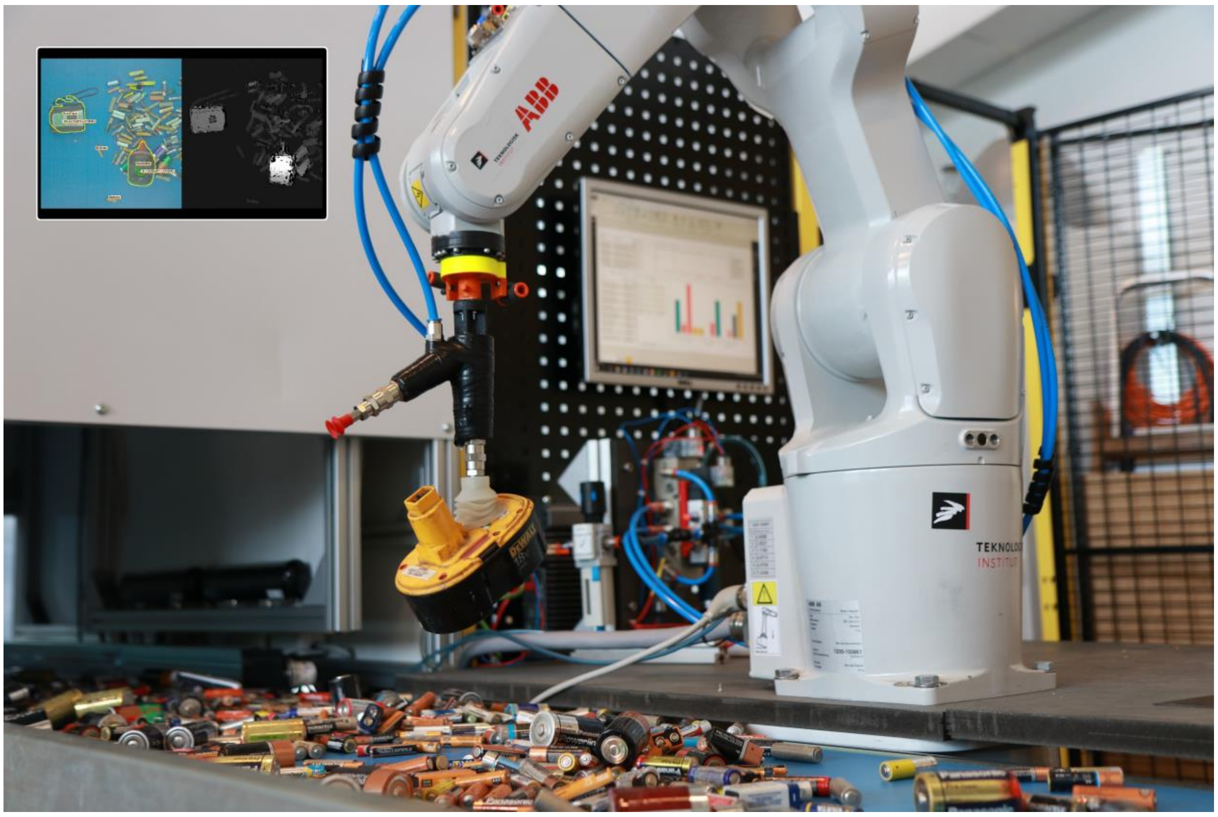
This made it easy to adjust the danger zones and optimize the speed settings through the intermediate points in the pick pattern to minimize decelerations.

9.1. Pick speed and success rate

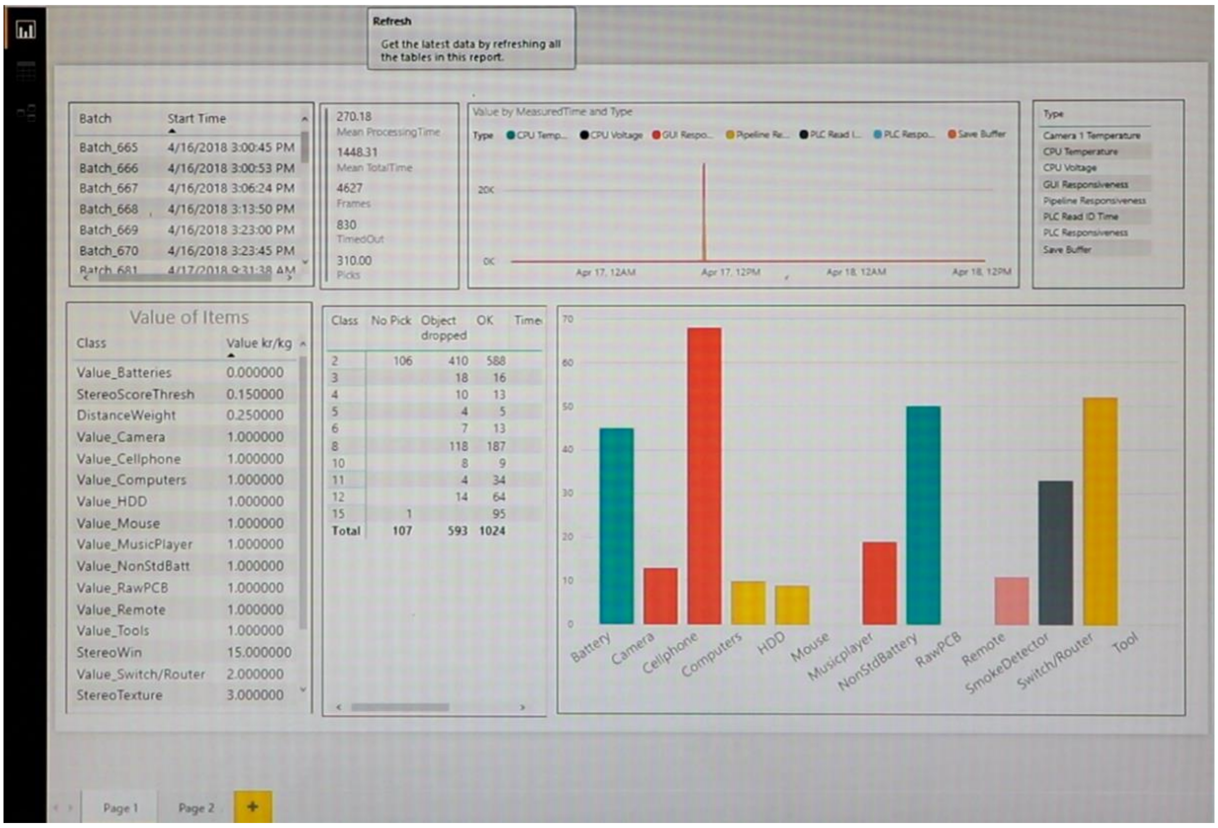
We found that pick speed and success rate was a big tradeoff.

Load	**Speed**	**Success**
No load	1.2 picks per sec	100%
Light items (remotes, cell-phones, light switches)	1.2 picks per sec	64%
Light items	0.8 pps	96%
Heavy (HDD, Large switch, Akku)	1.2 pps	8%
Heavy	0.8	23%
Heavy	0.5	76%
Heavy	0.25	90%

Also, some items from some surfaces were not possible to pick, even if the "delta Z" meas-ure from the stereo vision system was as low as others.



Final demonstrator



Refresh

Get the latest data by refreshing all the tables in this report.

Batch	Start Time
Batch_665	4/16/2018 3:00:45 PM
Batch_666	4/16/2018 3:00:53 PM
Batch_667	4/16/2018 3:06:24 PM
Batch_668	4/16/2018 3:13:50 PM
Batch_669	4/16/2018 3:23:00 PM
Batch_670	4/16/2018 3:23:45 PM
Batch_681	4/17/2018 9:31:38 AM

270.18
Mean ProcessingTime

1448.31
Mean TotalTime

4627
Frames

830
TimedOut

310.00
Picks

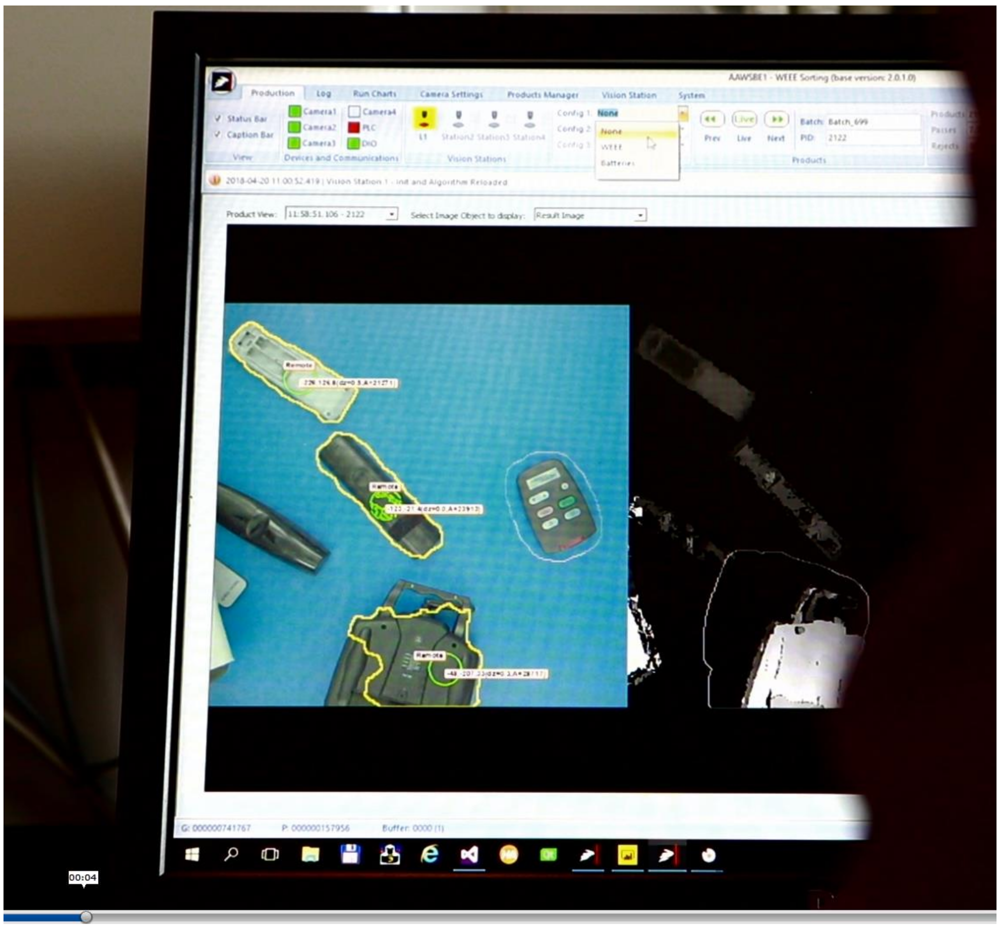
Type
Camera 1 Temperature
CPU Temperature
CPU Voltage
GUI Responsiveness
Pipeline Responsiveness
PLC Read IO Time
PLC Responsiveness
Save Buffer

Value of Items

Class	Value kr/kg
Value_Batteries	0.000000
StereoScoreThresh	0.150000
DistanceWeight	0.250000
Value_Camera	1.000000
Value_Cellphone	1.000000
Value_Computers	1.000000
Value_HDD	1.000000
Value_Mouse	1.000000
Value_MusicPlayer	1.000000
Value_NonStdBatt	1.000000
Value_RawPCB	1.000000
Value_Remote	1.000000
Value_Tools	1.000000
StereoWin	15.000000
Value_Switch/Router	2.000000
StereoTexture	3.000000

Class	No Pick	Object dropped	OK	Time
2	106	410	588	
3		18	16	
4		10	13	
5		4	5	
6		7	13	
8		118	187	
10		8	9	
11		4	34	
12		14	64	
15	1		95	
Total	**107**	**593**	**1024**	

The FAT demonstrator included built-in business intelligence.



Selection of picking case.

10. FAT

The final demonstrator was integrated in the DTI Vision Box 2.0 base software, where it was possible to select WEE or Battery demo, two different collection schemes, and the user could change the values of individual groups in an easy to use property sheet (pincode locked) and affect the order of which items was picked.

We noted recognition rates, pick success rate, and bin purity for both cases. They used two differently trained classifier caffeemodels with the following pixelwise training performance output:

10.1. Classifier data

The final dataset consisting of weee and images of lots of batteries (in the test set there are no images of wee on top of batteries, but the fcn models are also trained on that through random permutations of cutting out WEE and placing on different backgrounds).

The dataset used for the cnn model consist of the same images but cropped around the objects in the image.

Iteration: Cnn error rates

75000:9.486165820431506

35000:9.765624809265141

15000:9.683794275023828

7000:12.401574559024123

4000: 14.201183151850431

2000: 22.586872150832583

1000: 21.07355822915391

starts to overfitting between iteration 35000 and 75000. FAT uses a network that was saved somewhere between those iterations.

The raw pixel accuracy correlate too much with the number of background pixels (if 90% of the pixels are background we get 90% accuracy by always saying everything is background)

The accuracy we measure on the fcn models are therefore the mean accuracy

mean accuracy == sum(accuracy for each kind of object class)/number of object classes

- low res fcn

iteration: mean accuracy

10 000 mean accuracy 0.7827627656178594

20 000 mean accuracy 0.7976270811079043

40 000 mean accuracy 0.7874321551790433

80 000 mean accuracy 0.7696920777471312

58 0000 mean accuracy 0.7721369540058594

Conclusion: very fast learns what it can, not much overfitting after that either

- middle resolution fcn

10 000 mean accuracy 0.9055274819144579

20 000 mean accuracy 0.9047122293760249

80 000 mean accuracy 0.9135276036582057

160 000 mean accuracy 0.92201292645498

320 000 mean accuracy 0.9188935799157908

learning stops around 320 000 iterations

- high resolution fcn

160000 0.9446864988010792

80000 0.9444267057513225

40000 0.941488096023677

20000 0.9449047261732284

10000 mean accuracy 0.9435942082954838

The final mean accuracy 0.9446864988010792 of the system

the pixel by pixel accuracy of that network is 0.8654152863945057 which is caused by the fact that the trash category is really hard. The mean accuracy is much higher because the accuracy on the other categories is so much better.

10.2. Performance

WEEE

real/classified	Trash	PickObject	total	Success
Trash	487	13	500	**0.974**
Pick object	78	2966	3044	**0.974376**
total	565	2979	3544	

Batteries

real/classified	Batteries	PickObject	total	Success
Batteries	1602	0	1602	**1**
Pick object	37	1439	1476	**0.974932**
total	1639	1439	3078	

Pick

	Pick success	Dropped	No Pick	Total	Success
WEEE-One by one	488	6	4	498	**0.97992**
WEEE-Chaos	443	43	26	512	**0.865234**
Batt-Chaos	468	12	19	499	**0.937876**

Rate

	PPM
Light	48
Heavy	9

Output Bin

	Go through	#2 robots (*1)	#3 robots	Bin 1	Bin 2	Bin 3
WEEE	17.0%	2.9%	0.5%	98.6%	99.1%	100.0%
Batteries	11.0%	1.2%	0.1%	100%	100%	100%

(*1) Or re-iterating on the same system. Stena goal is 3%

Follow up test to validate results are repeatable: 50-50 WEEE / BATT case

missed	13	
missclass	7	5 was on big trash could not be picked
total	102	2 could be picked and ends up in wrong bin
bin purity:	***99.3%***	
go through	***14.7%***	

10.3. User acceptance

The machine vision robot sorter equipment successfully identified and sorted a stream of mixed used electronics and batteries. Overall impression was positive, National Technical Manager Bent Pedersen said that it is a very novel approach and technology readiness looks close to production ready. They appreciated the ease of configuring the picking strategies and the display of graphs in Power BI, in particularly histogram of object types, and pick success per object. An impressive perspective was that the same system has potential to work with different cases, based on the training and pick parameters.

In terms of risks: They could be concerned about the deep learning teaching – do we teach it the correct things? Also, the classification is based on derivative reasoning: it does not detect the material, or presence of a material or battery inside, it selects it based on "it looks like something we know to have this material/battery inside".

What about the idea of presorting into uniform groups of equipment, such as copper motors in this pile, cell phones in this pile, hard drives here, teddy bears here, etc.? That is pretty much as they do by hand at technoworld but it would require a much larger load than they currently have, plus, the design of the next step, reaping the benefits of custom handling of the uniform output fractions would require a new project per group.

So unfortunately, they were not too hyped in the case of WEEE presorting, as they do not handle that much WEEE in Denmark. What they cannot handle is just sent to Technoworld Sweden.

Instead, during the meetings the end users brainstormed and prioritized other waste fractions where they could see the demonstrator have more immediate utility:

1. infeed to shredder sorting systems. recognize damaging objects. costs 500.000 DKK each time something obstructs the shredder. This could be a vision module and a laser pointer that points out tings for an operator if the robot cannot pick the large objects.

2. Copper/Brass , motor fittings. Hyperspectral cameras are not good for metal, so the deep learning approach may be better.

3. Battery presporting (only in vissenbjerg) they are really keen on that. want to see it working for that.

4. Spray cans , want to detect foam cans, Plastic Caps, loose plastics. Id unit with hyperspectral upgrade is a good solution for that.

5. WEEE - not so much, as they lack volume and dont know much about it. Might become more relevant if they win the DANWEEE (who went bancupts) market 15000tons per year. Initial plan will just copy-paste from stena technoworld sweden.

6. Carbon from newspapers, but this application is too fast for the robot system.