SortX

User guide

Cerrahoglu Ali
Duan Boyan
Domanski Lukasz
Dilyana Karavasileva
Daniel Man
Pandey Abhishek
Sekuloff Ruth

1. Introduction

This document describes what the systems consists of, the process of installation and setup of the robot, as well as procedures for its safe and efficient use.

This document consists of the following sections:

2. System Description
Outlines the different components of the system and describes the hardware.

3. Setting up the system
Includes instruction on how to get started. Read this section carefully and follow the steps as outlined. This will ensure a normal operation of the whole system.

4. Using the system
Includes a description of all procedures required for the safe use of SortX hardware and software.

5. Troubleshooting
Provides solutions to the most commonly occurring problems.

2. System Description

2.1 Key features

SortX is a prototype of a robot, assisting warehouse workers. It is capable of efficiently stacking boxes on pallets in preparation for inter-fulfillment shipping. To achieve that, the robot:

- Detects a set of coloured boxes using its vision system.
- Computes a way of stacking those boxes on a pallet so that the space on the pallet is used as efficiently as possible.
- Picks the boxes and leaves them on the defined by the algorithm destination.

2.2 System composition

The system is composed of:

A robot (hardware module) with the following components**:**

- Crane: A structure built with LEGO pieces that supports the moving mechanism of the robot.
- Three-axis robotic gantry: A bridge-like structure located on the top of the crane, a moving part of the robot. It allows for movement of the grabber in three dimensions.
- Grabber: A mechanism capable of picking up and rotating boxes.
- Camera: A webcam located at the top of the crane that is used for box detection.
- EV3 bricks: The “brain” of the robot, a computer that receives inputs from sensors, powers and controls motors.
- Working Area: The wooden structure underneath the robot. Red tape marks borders of the working area - this is how far the grabber can move. The area is divided into two parts: Pallet Area and Pickup Area. The user places boxes for the robot to sort in the Pickup Area. The robot picks up those boxes and places them on the Pallet Area.

2.2 System composition

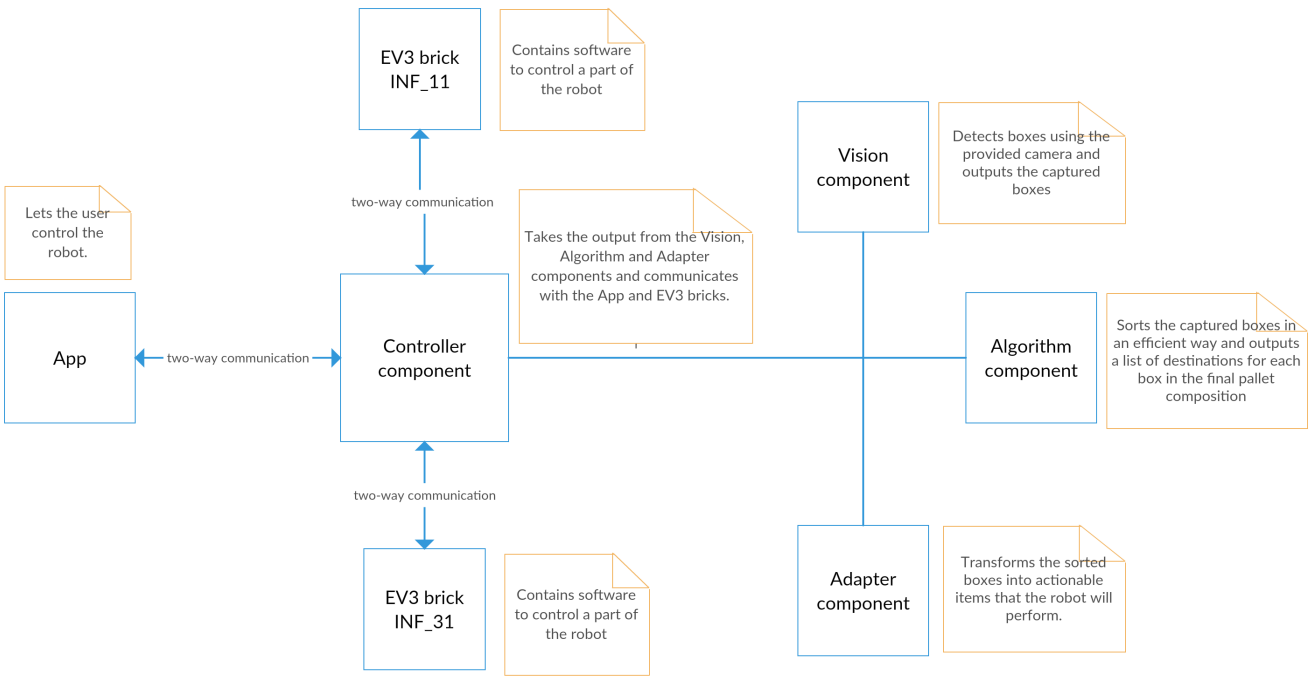
A software module with the following components:

- Vision component: Detects boxes using the provided camera and outputs the captured boxes with bounds produced by the vision system.
- Algorithm component: Sorts the captured boxes in an efficient way and outputs list of destinations for each box in the final pallet composition.
- Adapter component: Transforms the sorted boxes into actionable items that the robot will perform.
- Controller component: it acts as a bridge between App - (Vision, Algorithm, Adapter) - EV3. It takes inputs from the App and then calls the above mentioned modules. It sends actions to the EV3s to perform.
- Starting script: Starts the controller and listens for messages from the app. This script bootstraps the controller initialisation.

An app which is the main source of communication between the controller, software system, and EV3software.

- Dashboard page: This page let the user interact with the robot and software system.
- Settings page: The user can set up the IP of the EV3s and the computer that runs the starting script.

Note: As shown in the control diagram the App is connected to the controller component using bluetooth. This means that the robot can be controlled through any computer that the app is installed on as long as it is connected to the controller component. This gives the user the flexibility to install the app on a mobile device like a tablet computer and still control the robot.



3. Getting started

3.1 Requirements

To run SortX software, a computer with Bluetooth connectivity is required. Software can be used on any Linux distribution (such as **DICE** or **Ubuntu**).

3.2 Installation

Before doing everything else, you need to get your computer ready for running SortX software. This section explains how you can do this in details.

Installation of dependencies

Open up a terminal window and run the following commands. This will install Python 3, pip, and mosquitto (include links for each).

```
$ sudo apt-get update
$ sudo apt-get install python3 python3-pip mosquitto
```

Using pip, install the software dependencies:

```
$ pip install pip --upgrade
$ pip install opencv-contrib-python numpy matplotlib scikit-
learn scipy paho-mqtt
```

Obtaining necessary code

The necessary software is available as a GitHub repository at https://goo.gl/d9cr8j. After you access the repository, press the green button Clone or Download and then Download ZIP.

Once the zip file is downloaded, navigate to the directory you want to put the code and unzip the downloaded file using the following commands in the terminal.

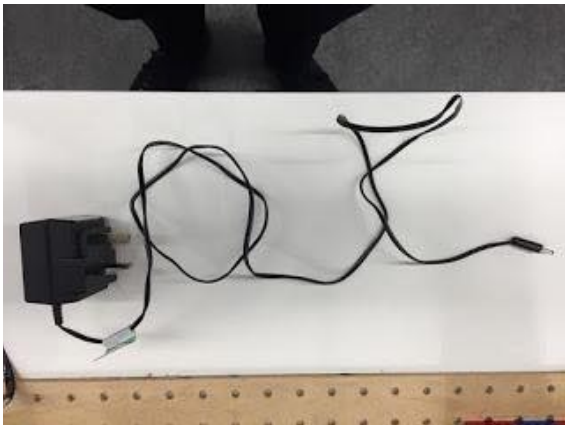
```
$ unzip system-design-master.zip
$ cd system-design-master
$ echo “PATH=${PATH}:$(pwd)” >> ~/.bashrc
$ source ~/.bashrc
```

3.3 Hardware setup

The robot is shipped pre-built and the hardware is ready to be used “out-of-the-box”.

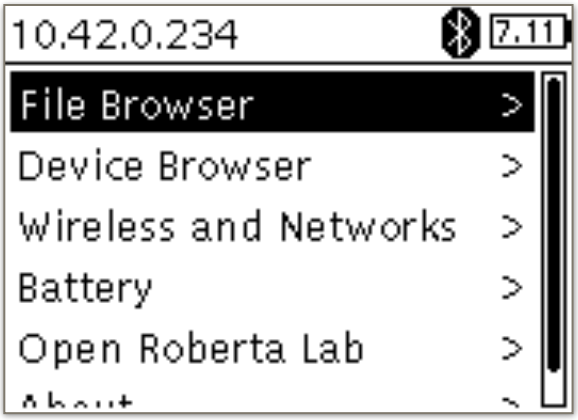
Robot Power Supply

The use of batteries are not needed since it is a stationary robot. A charger is used to charge the EV3’s. The images below show the charger itself and the port at which you need to insert the charger wire into.



Bluetooth connection on EV3

The computer you use needs to have an active bluetooth connection to both EV3 bricks because the controller software will communicate with the EV3 brick software. Follow this tutorial, section 2.2. It describes how to establish a bluetooth connection between a computer using Ubuntu and an EV3 brick. Once the devices are connected, the IP of the EV3 bricks will be shown in the top-left corner of the screen.



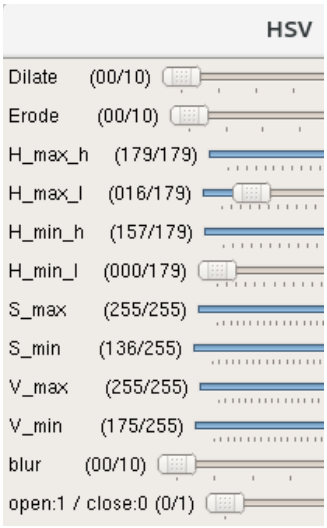
3.4 Setting up the vision module

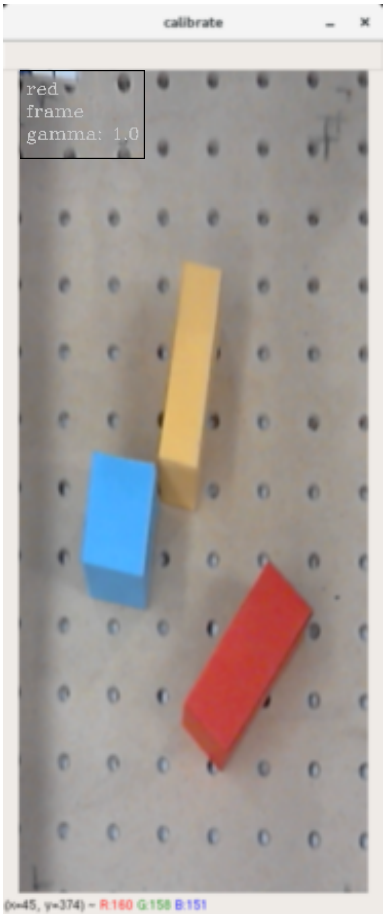
The vision system of the robot uses colours to detect and localise the boxes accurately. Since different lighting conditions introduce variation in detection of colours, the robot must be calibrated for the lighting conditions in its current environment. Consequently, any significant changes in the lighting conditions will result in a need for re-calibration.

Plug the vision camera USB into the computer. To start the calibration run the following command in terminal:

```
$ ./Calibrator.py
```

This will bring up 2 windows, one showing the parameters available to calibrate and another showing a live stream from the robot's camera. (shown in the Figure)

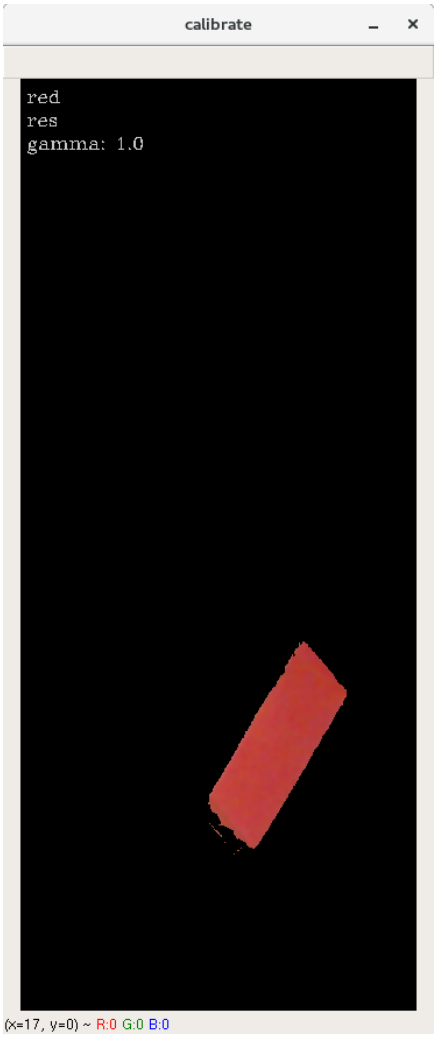


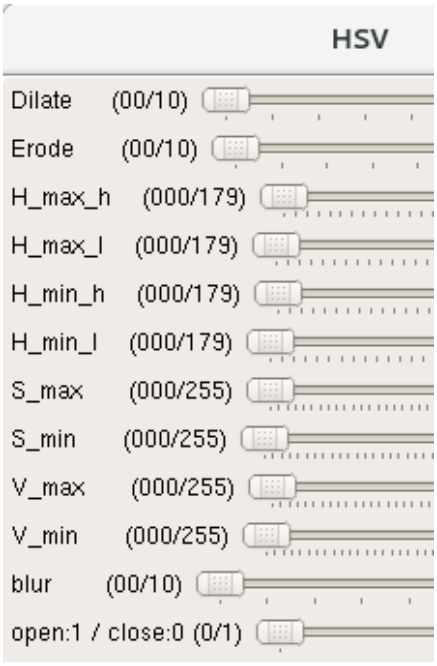


The Calibration procedure starts by choosing a colour mode which represents the colour in which we are calibrating for and is shown at the Top-Left corner of the video stream. Initially a set of colours are present to calibrate which are shown in the documentation of the file but more colours can be added on the go.

When any one of the windows is selected, pressing the button 't' can be used to change the current colour mode and '<' & '>' keys can be used to toggle the display mode which represents the kind of video feed(topleft corner of figure). By default, the display mode is set to 'frame' which shows the unprocessed image, but for calibration purposes you may want to move to 'res' display mode which shows the effect of the selected parameters on the original image.

3.4 Setting up the vision module





The Calibration process is essentially finding a range for the Hyper parameters H,S,V for each colour that identifies a colour uniquely. This may look like a hard task but knowing what the H,S,V values represent makes the process very intuitive. The 'H'(0-179) value sets the target colour, for eg. choosing (0-20) for H normally gives you almost all the shades of red colour and remove all other colours. The 'S'(0-255) value sets the saturation or the purity of the colour. For e.g, yellow contains a lot of red and can be removed from the image by increasing the saturation when finding the red colour. The 'V'(0-255) parameter represents the brightness of the colour. Setting a high range(150-255) will filter out any dull shades and keep the bright colours. Refer to this document to know more about the HSV colour space. As seen in the figure, The H value is controlled by 4 parameters. The purpose of the 4 values is to create a flexible range of colours. For e.g, When finding a red colour, some shades are present in the range 160 -179 and most are between 0-20. So in order to to find the all the red shades in the image and remove any other colour, we create a range from 0-16(H_min_l to H_max_l) and 160-179(H_min_h to H_max_h). For most other colours, set both H_min_h and H_max_h to 179.

The parameter Blur controls the level of blur applied to the image. Other parameters provide more control on the detected boundary and normally will not be used during calibration. To know more about these parameters read this document.

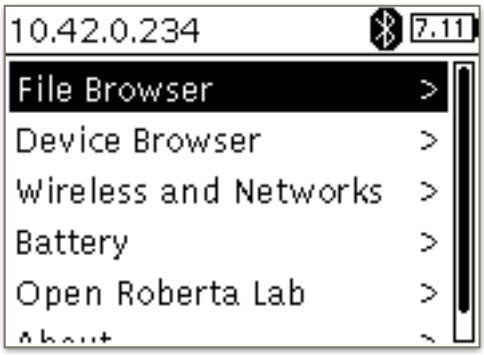
Note: Make sure to keep the boxes of all the other colours present when calibrating for a colour. This will ensure that no two colours are detected together by the values chosen.

The above figure shows the result of a calibration for 'red' colour in an image. After calibrating a color, press ('c') this will save the configuration for that colour. Then the colour mode can be changed by pressing ('t') to move to a different colour. After completing the calibration for each colour, press 's' to save the parameters into the system and exit.

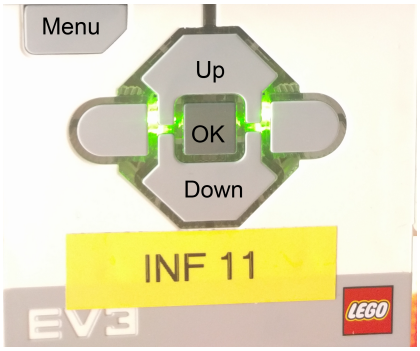
3.5 Software setup

Starting the StartEV3 script on the EV3 bricks
You need to start the software script on both EV3 bricks.

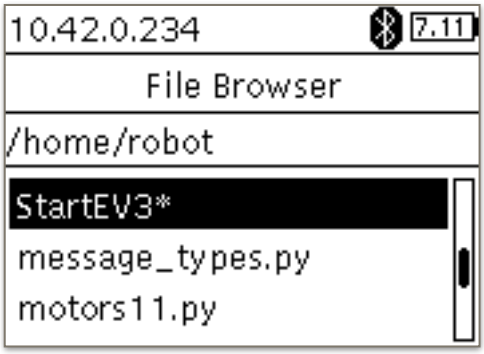
- Copy down the IP in the top left corner of the screen (for both the EV3 bricks): you will need these IPs later.
- Click on the File browser (clicking OK button)
- Scroll down (Up/Down buttons) until you see StartEV3 and click it (OK button). This will run the starting script which loads each component in the robot software.



EV3 brick screen with File browser



EV3 control buttons



EV3 brick screen with the starting script

Starting the controller
On the computer that you downloaded the source code, run the following:

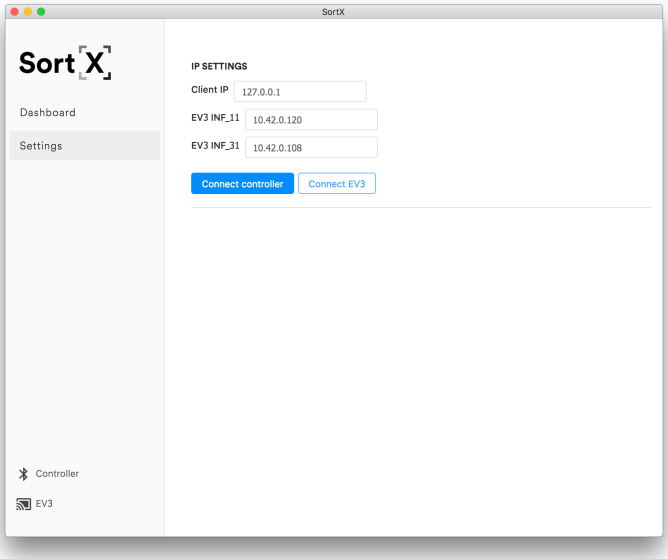
```
$ ./start.sh
```

3.5 Software setup

Starting the app

During this installation we will assume that the app is present on the same computer as the controller component:

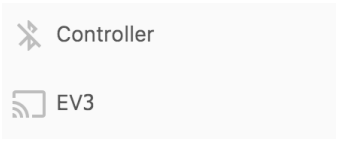
- Start the app by opening /app folder in the / sortX directory
- Click the AppImage icon. This will open the app
- Go to the settings tab in the navigation bar
- In the Client IP input, write 127.0.0.1 (this means that the app runs on the same computer)
- In the other two inputs write down the IP of both EV3 bricks
- Click on Connect controller button. This will instantiate a connection between the app and the controller
- Click on Connect EV3 button. This will instantiate a connection between the controller and the EV3 bricks.



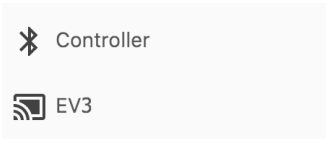
Settings page, with all the connections established.

Connectivity status:

The app shows the connectivity status for the app - controller connection and the controller - EV3 connection. You can see this at the bottom of the navigation bar.



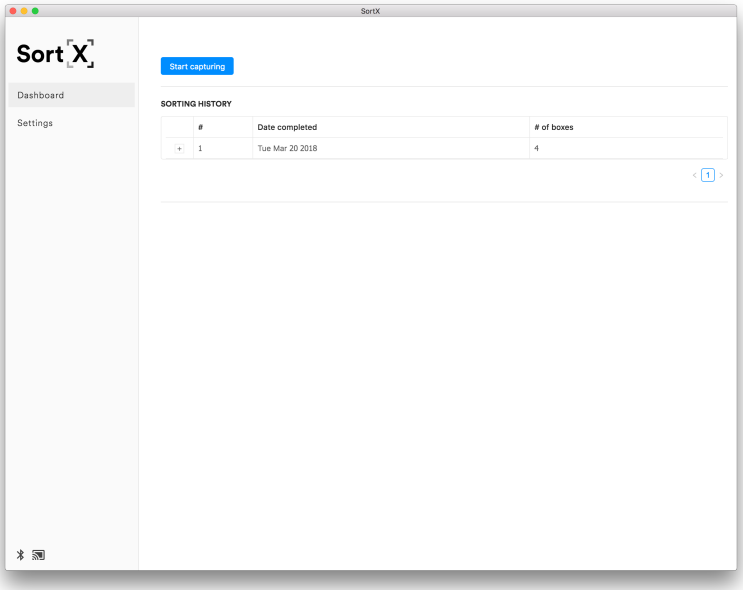
Not connected



Connected

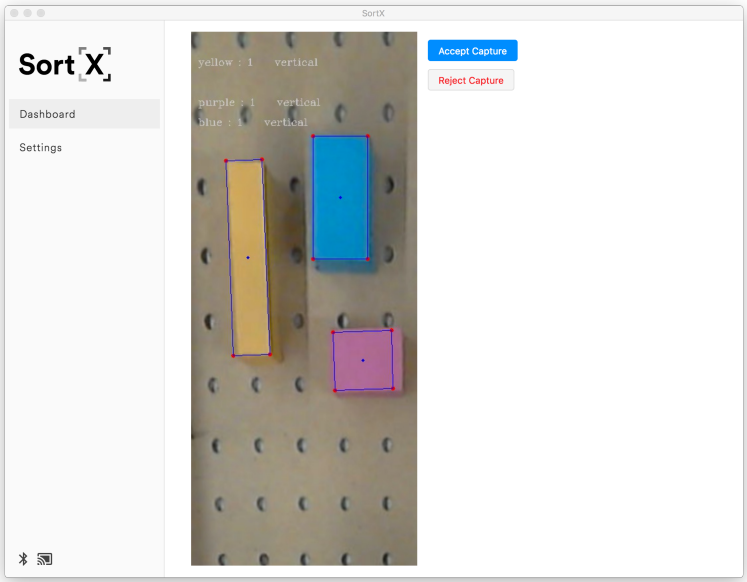
4. Operation

4.1 Sorting boxes



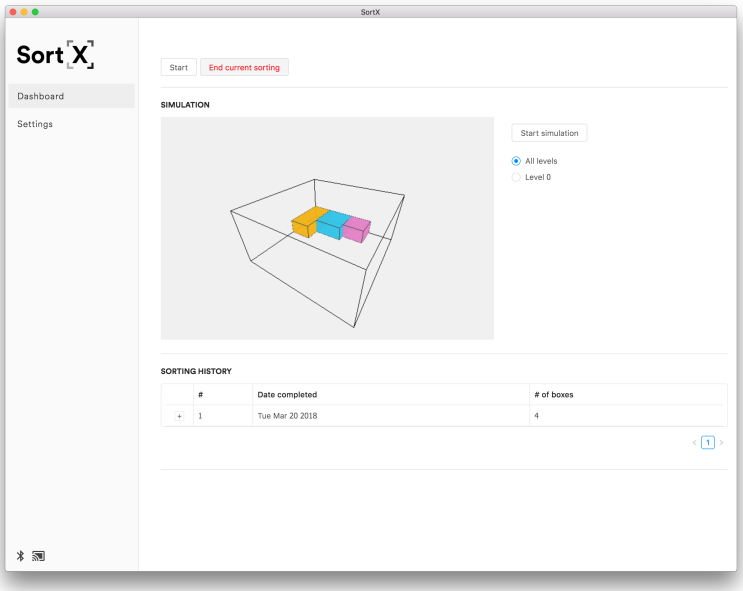
1. Click Start capturing button: this will start a video feed from the camera with the boxes in the pick area.

2. Put the boxes you want to sort until you see them in the video feed from the app. The vision system puts bounding boxes around each detected box.



3. Click Accept capture button: this will instruct the controller to start the algorithm module and sort the boxes. (if you press Reject capture, the page will reset to its initial state where you can start another capture phase)

4. Click Start button: this will instruct the robot to start moving the boxes to the pallet area



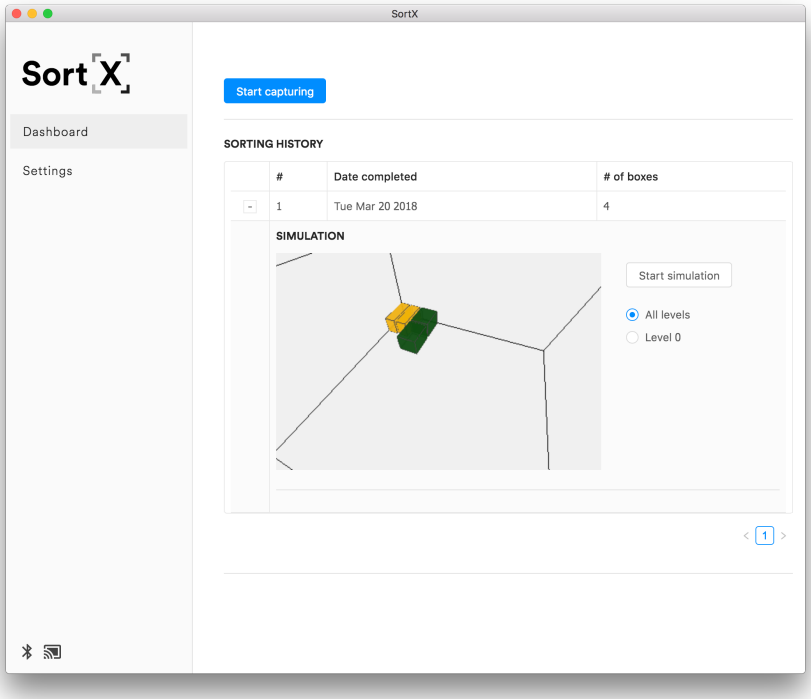
5. While the motor is sorting the boxes into the pallet area, you will see the Start button having a loading state. When the robot is done sorting the boxes in the pallet area, the dashboard page in the app will return to the initial state.

4.2 Viewing the history of sorted pallets

1. You can view the history of sortings in the table below the control buttons (Start capturing button)

2. Each sorted pallet that was stacked previously is shown in the table as a row. You can view the date that the process ended, as well as how many boxes it sorted

3. You can expand each row by clicking on the + button on the most left-side column of the row. This will show you the 3D simulation of the boxes that were sorted at that process. If the boxes were sorted into more than 1 layer, you can highlight each layer in the simulation by clicking the radio buttons

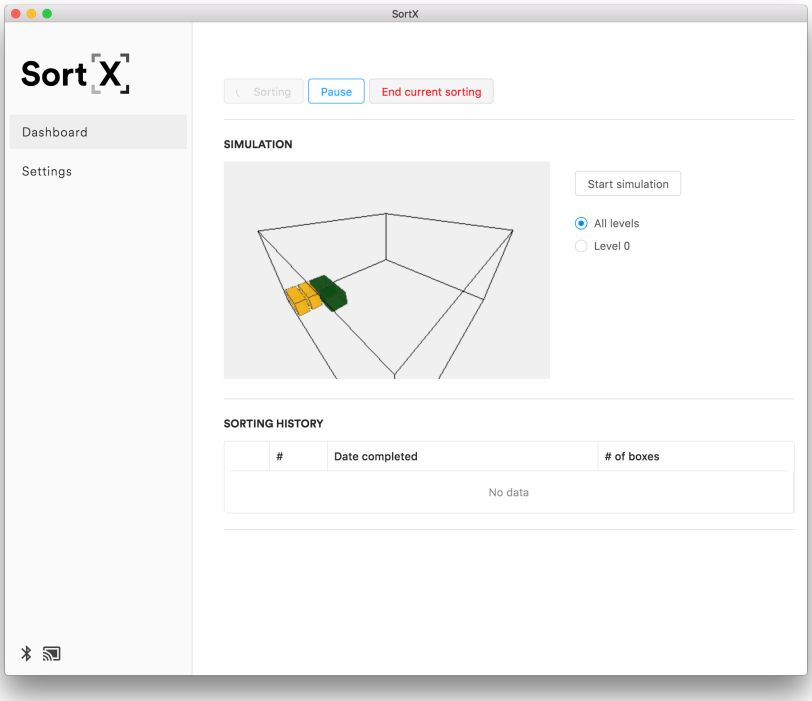


4.3 Actions that can be performed while sorting

1. You can view the 3D simulation of the boxes that the robot is sorting at the moment

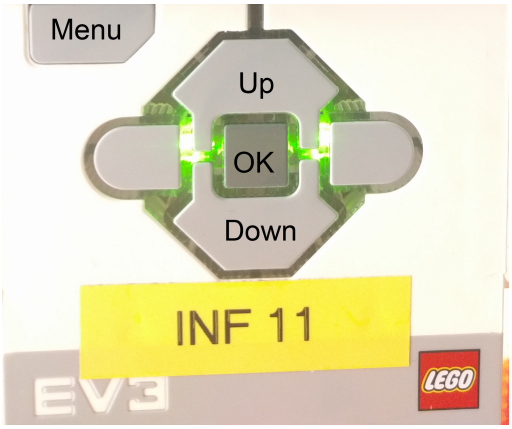
2. While the robot is moving the boxes into the pallet area, you can pause and resume the actions. If you click Pause button, the robot will stop and wait for resume

3. You can also end current sorting in the case of an error. Ending the current sorting will delete all the actions that were queued for the robot to perform and put you into the initial state.



4.3 Powering off the system

1. Close the app
2. Close the terminal running the start.sh script (the controller script)
3. On both EV3 bricks, press "Menu" button
4. On both EV3 bricks, select “Power Down” by using “Up” and “Down” buttons
5. On both EV3 bricks, press “OK” button



EV3 control buttons

5. Troubleshooting

This section is divided into three sections: Installation, Software and Hardware. If you experience a problem, identify which part it corresponds to and to try the suggested solutions.

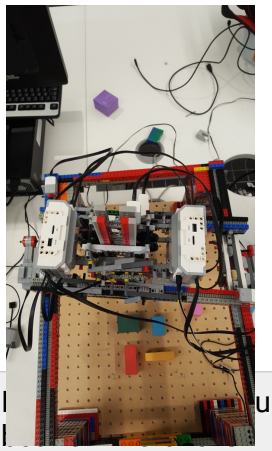
5.1 Installation Troubleshooting

Symptom	Cause	Solution
Sudo error on DICE machine: “user is not in the sudoers file. This incident will be reported.”	User doesn’t have root access on the DICE machine.	Email Garry Ellard (gde@inf.ed.ac.uk) about requesting root access onto a machine to be able to run “sudo” commands.

5.2 Software Troubleshooting

Symptom	Cause	Solution
Grabber is attempting to pick up boxes from incorrect positions.	Position of boxes may have changed after the capture of boxes was accepted. Vision system may not have been calibrated or the lighting environment has changed.	End this sorting session and start it from the beginning again. If it happens again, calibrate the vision system. (Section 3.4.1)
Video preview in the application shows that some or all of the boxes are not correctly bounded.	Vision system may not have been calibrated or the lighting environment has changed.	Try to move the incorrectly captured boxes and see if this solves the problem. If not, reject the capture. If boxes are still not recognised, calibrate the vision system.

5.3 Hardware Troubleshooting

Symptom	Cause	Solution
The movement of the robot is chaotic or it gets "stuck" but there is a correct simulation in the app.	One of more motors may have been accidentally disconnected. They may have not be connected with the correct cable. One or more motors may be malfunctioning	End sorting in the app. Exit the terminal that runs the start.sh script. Turn the EV3s down and follow the labels on the cables to check that all connections are correct. Turn the EV3s on. If the problem still exists, replace the motor/s with a new one.
Motors on axis X or Y are misaligned. 	Motors on the same axis may be misaligned by accident. One of the motors may not be working.	Power the EV3's down and manually adjust the position of the structure. Turn the EV3's on again. If the symptom happens again, repeat the solution for the above symptom.
I urn on one or I	Since batteries are not used, the EV3 brick/s might be malfunctioning.	Check that the EV3's have been put on charge. If so, change the EV3 brick/s with a new one.

6. Appendix

6.1 Additional reading

For more information on git and GitHub, you can refer to this introductory video guide: https://www.youtube.com/watch?v=0fKg7e37bQE

More information about terminal command pip can be found in it's official documentation: https://pip.pypa.io/en/stable/

More information about using sudo command can be found here: https://wiki.debian.org/sudo

Additional reading about Morphological Operations : https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html

Learn about HSV Color Space : https://en.wikipedia.org/wiki/HSL_and_HSV

6.2 List of cables connections

Device	**Cable**	**EV3 port**
Motor M2	31/A	Brick INF31, port A
Motor M1	31/B	Brick INF31, port B
Motor M6	31/C	Brick INF31, port C
Motor M5	31/D	Brick INF31, port D
Sensor S1	31/1	Brick INF31, port 1
Motor M3	11/A	Brick INF11, port A
Motor M4	11/B	Brick INF11, port B
Motor M7	11/D	Brick INF11, port D
Sensor S2	11/1	Brick INF11, port 1
Sensor S3	11/2	Brick INF11, port 2