Technical Report

Joel Fåk, Viktor Eriksson, Per Boström, Eric Gratorp,

Astrid de Laval, Sven Ahlberg

Version 0.1

Status

Reviewed		
Approved		

Images and Graphics, TSBB11 2012-12-14

cdioproject3@googlegroups.com

PROJECT IDENTITY

CDIO, Group 2, Identification of LCD monitors
Linköping University

Name	Responsibility	Telephone	E-mail
Joel Fåk	Project manager	073-425 86 68	joeka281@student.liu.se
Viktor Eriksson	Document manager	070-468 07 15	viker942@student.liu.se
Eric Gratorp	Test manager	070-899 25 42	erigr222@student.liu.se
Sven Ahlberg	Design manager	076-127 06 78	sveah621@student.liu.se
Per Boström	Quality manager	073-248 58 59	perbo612@student.liu.se
Astrid de Laval	Scrum master	070-119 21 69	astde117@student.liu.se

Group mail: cdioproject3@googlegroups.com

Customer: IEI, Linköping University, 581 00 LINKÖPING,
Contact person: Kristofer Elo, kristofer.elo@liu.se

Table of Contents

Document History

Version	Date	Performed changes	Performed by	Reviewed
v0.1		First draft	EG, PB, SA, JF, AdL, VE	

Introduction

2 million LCD monitors & TVs are sold each year in Sweden. As a monitor contains both environmentally harmful substances and valuable materials, the interest in an automated recycling process is large. This project is a part of the research project HÅPLA which investigates the possibility for such processes and is performed as a part of the course TSBB11[1] Images and Graphics at Linköping University.

The objective of the project is to automatically detect and locate LCD monitors in a pallet, pick them up and place them by the side of the box. This will be done using a Kinect camera and a Yaskawa SDA robot. The document provides an overview of how the system is constructed.

System Overview

The system consists of a sensor, a computer and an industrial robot. The sensor is a Microsoft Kinect camera that provides RGB images and depth images, placed over the robot with a good view of the pallet. The data from the sensor is processed in a computer to detect monitors and select which one to pick up. The computer then tells the industrial robot how to move in order to go to the selected monitor, pick it up and place it by the side of the box.

The industrial robot that the system is using is a Yaskawa SDA10 robot which is controlled by the control system Yaskawa NX100. The deployment diagram in figure 1 is an overview of how the parts are connected. In figure 2, image of the different components of the system are displayed.

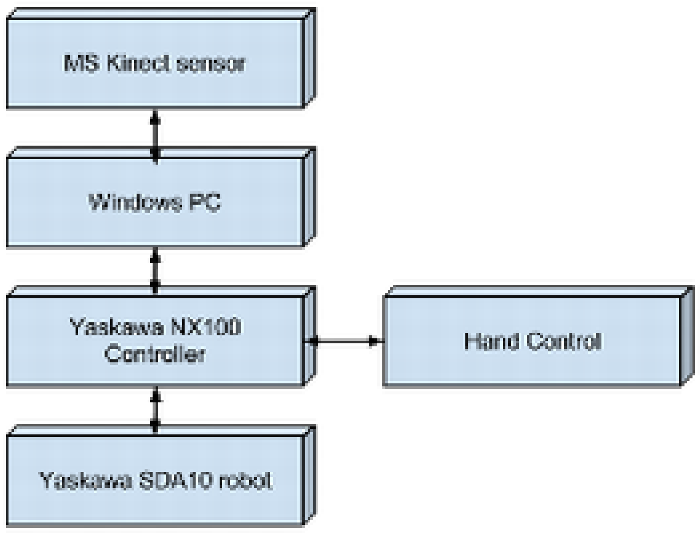


Figure 1: Deployment diagram

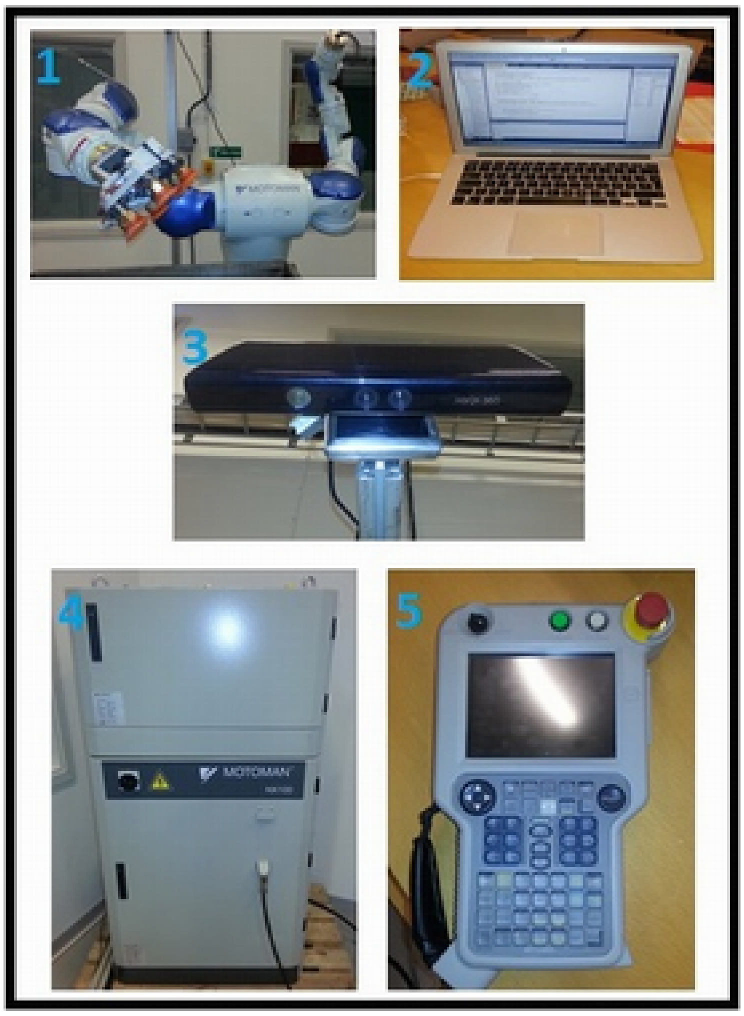


Figure 2: System components

1: The industrial robot, Yaskawa SDA10,
2: Computer,
3: Microsoft Kinect sensor,
4: Robot control system, Yaskawa NX100,
5: Hand controller

Kinect Interface

Image data from the Kinect sensor needs to be imported to the computer. To do this, the library OpenNI[2] with OpenNI grabber is used. The OpenNI Grabber requests different types of data streams from the Kinect camera. For our purpose it is enough to use the RGB image and the three dimensional point cloud. These two pictures are saved to files every time a new pick up cycle or calibration cycle is started.

Calibration

There is a small displacement between the Kinect depth sensor and the Kinect RGB sensor. This registration is automatically handled by the Kinect, which brings the range data in alignment with the colour data.

To be able to move the robot arm to a certain point in space corresponding to a point in the camera-centered coordinate system, a mapping between the image coordinate system and the robot coordinate system needs to be performed. This is done by moving the robot arm along a route of preselected points in the robot's world coordinate system. Using the data from the Kinect sensor, the system is able to find the corresponding points in the camera coordinate system. With the use of these point-pairs the calibration can be performed using an estimation of the rigid transformation corresponding to the rotation and the translation between the two coordinate systems.

The robot's grasping tool consists of six plastic suction cups placed in a larger circle. The center point of the larger circle is called the tool center point (TCP) and the current position of the TCP can be read from the control system.

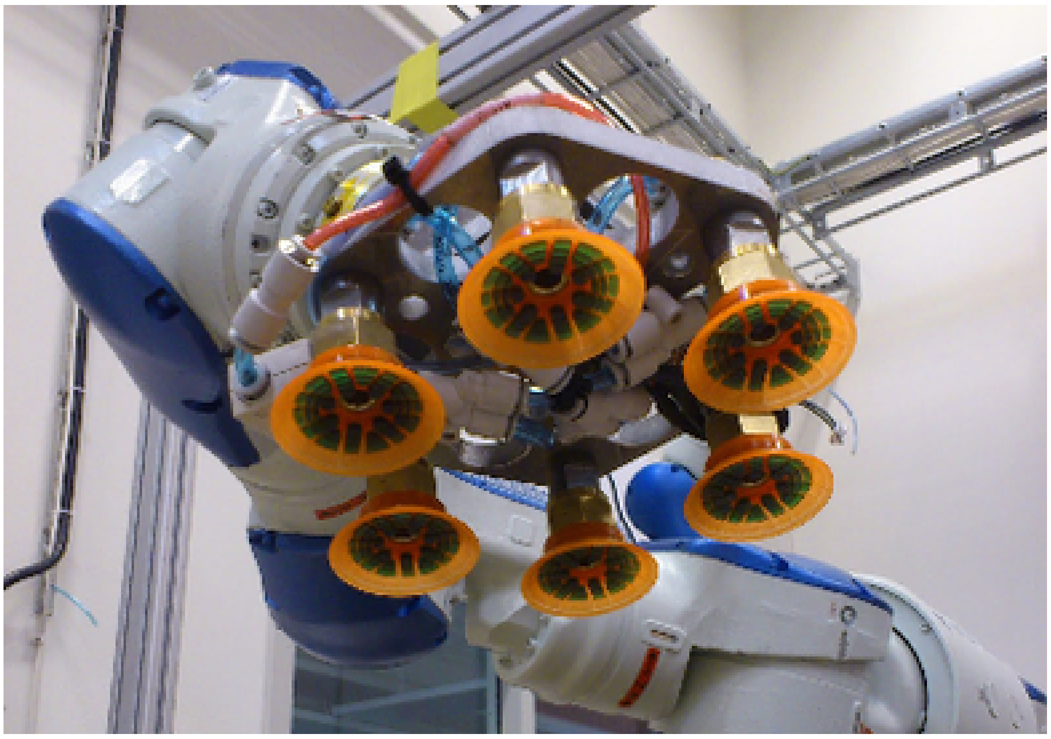


Figure 3: The robot's grasping tool.

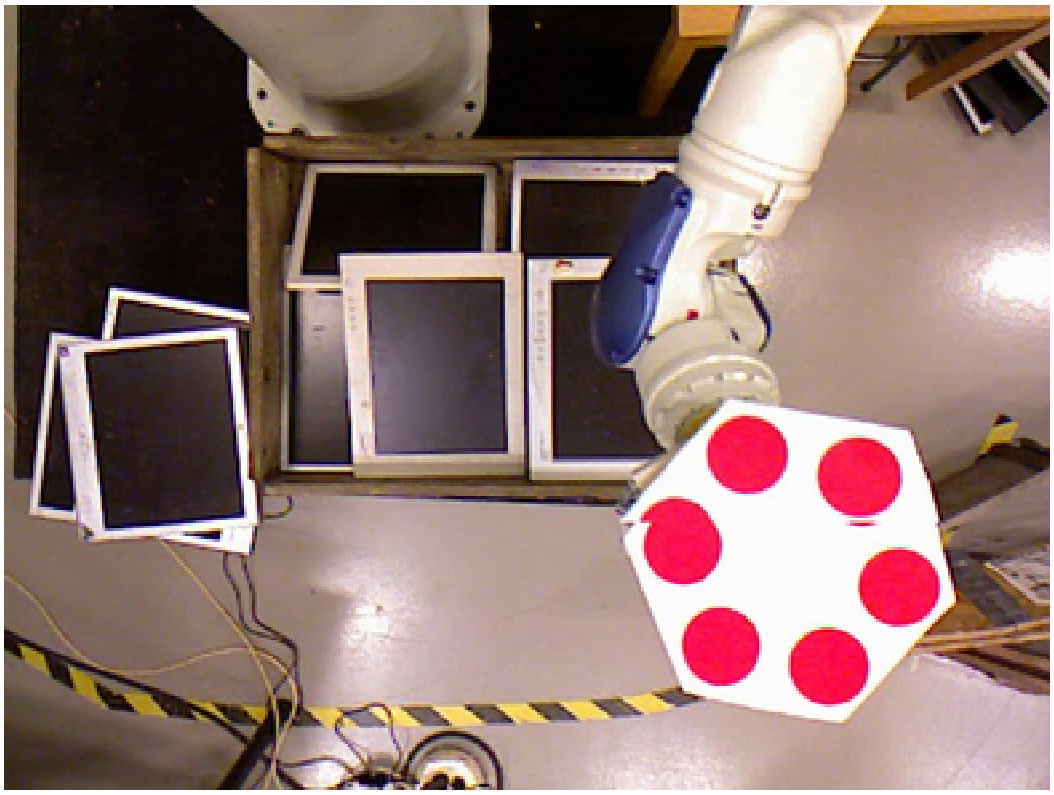
By detecting the grasping tool in the RGB image from the Kinect sensor, the position of the TCP in Kinect coordinates can be estimated.

Detection of Grasping Tool

During the calibration a paper with red circles corresponding to the plastic circles is placed on the grasping tool, as seen in figure 4 below.

Figure 4: The robot's grasping tool with a paper with red circles mounted for better detection.

Since the circles are red, they show up clearly in the red channel of the RGB image. This is utilized by subtracting the blue and green channel from the red channel, making the circles appear even more clearly.



To detect the circles, Canny edge detection[8] is used to find the edges in the image. As can be seen in the figure below, the edges marks out the circles clearly.

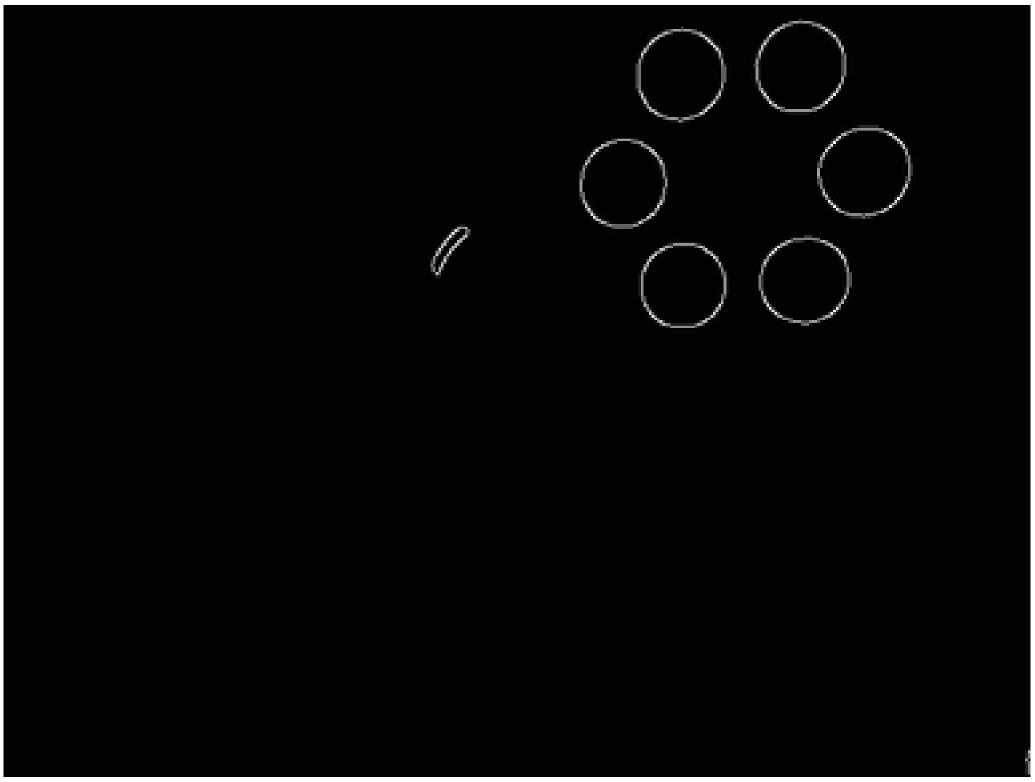


Figure 5: Kinect image after Canny edge detection.

The Hough transform[7] is basically a transform that can be used to detect circles in an image. Hence, it is perfect for our purpose. The transform detects the circle centers at the correct positions, see figure below.

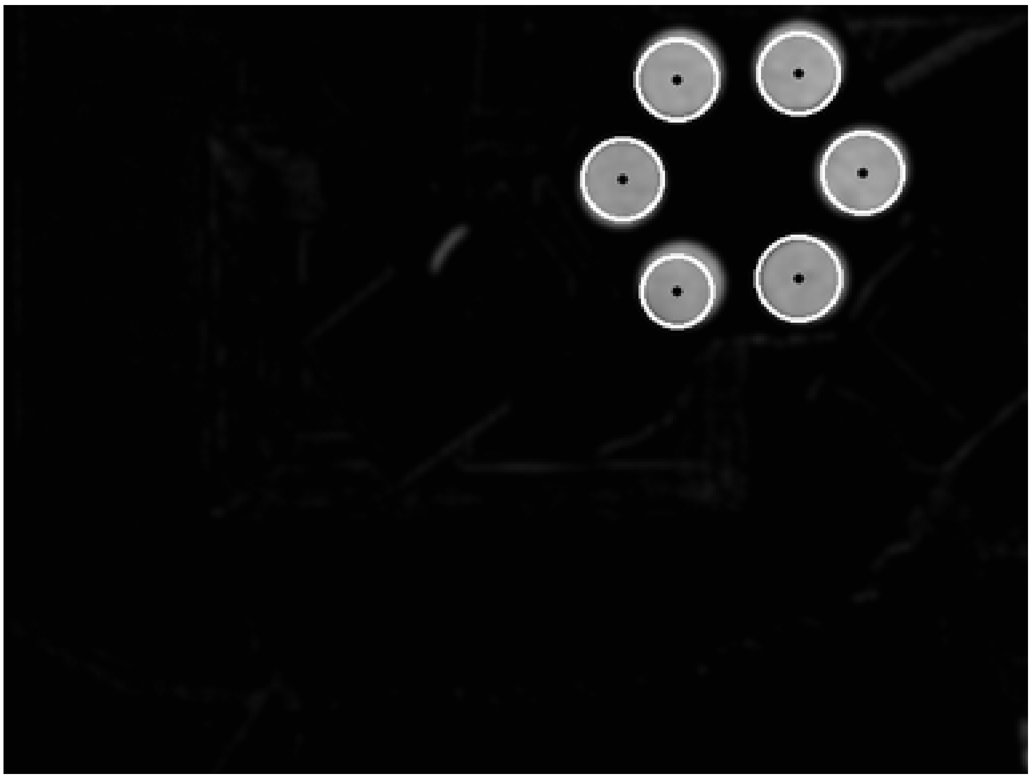


Figure 6: The grasping tool is the detected circles in the Kinect image. The image is the red channel minus the blue and green channel from the RGB image.

The RGB image provides a 2D-coordinate of the TCP, but since the world is in three dimensions this is not enough to map between the coordinate systems. The Kinect sensor also provides a point cloud, which makes it possible to pick out all 3D-points within the detected circles. Since the circles are drawn on a paper, all points should lie on a plane. By fitting a plane to the 3D-points within all circles, false circles, i.e. circles that do not correspond to circles on the grasping tool, can be removed. With the false circles removed, a 3D center point for each circle is calculated as the centroid of the points within the circle that also are inliers in the estimated paper plane. Since the number of circle inliers is different for each circle, it is necessary to calculate the centroid for each circle and then estimate the TCP as the centroid of all circle centers. If the TCP would be estimated as the common centroid of all inlier points, it would end up closer to circles with many inliers than circles with only a few inliers.

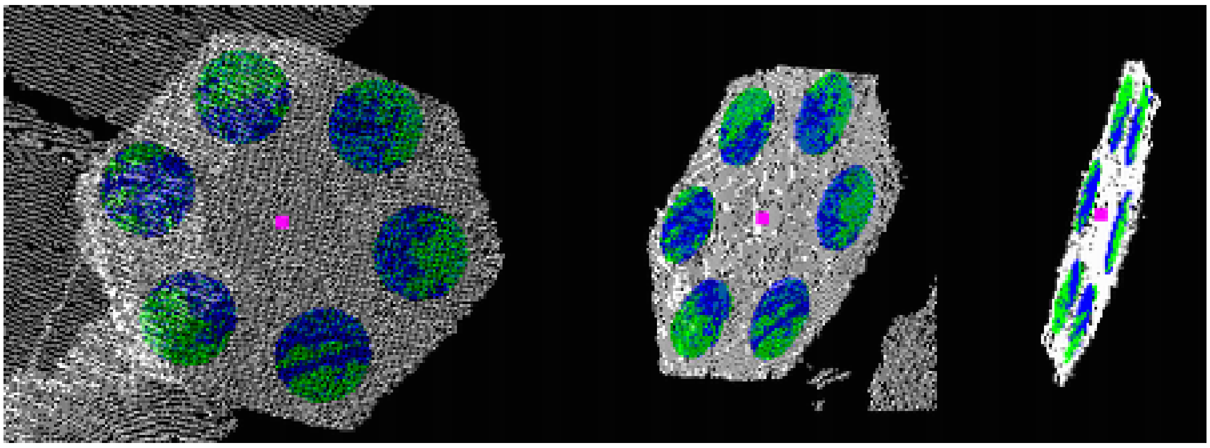


Figure 7: Point cloud from the Kinect sensor with the grasping tool detected. The green points are all points within the circles, the blue points are the inliers in the plane fitted to the paper with circles. The magenta point is the estimated TCP.

The procedure is performed four times in order to obtain a number of measurement data for every TCP position. This is done mainly because the Hough transform sometimes detects false circles that lie on a plane that is not the grasping tool plane, leading to an estimated TCP very far from the correct position. To remove these false TCP estimates some filtering has to be done. First a range filter is applied, which removes the points that are too far away from the Kinect sensor. This can be done since the calibration route is preselected and hence the maximum distance between the Kinect sensor and the TCP can be estimated. The points removed by the range filter are typically points on the floor next to the pallet. To increase accuracy even more, a radius outlier removal filter is applied. The filter iterates once through the entire input and for each point retrieves the number of neighbors within a certain radius. The point is considered an outlier if it has too few neighbors. The figure below helps to visualize what the radius outlier removal filter does. If the minimum number of neighbors is set to 2 and radius to r, the blue point will be removed and the green point will remain in the data set.

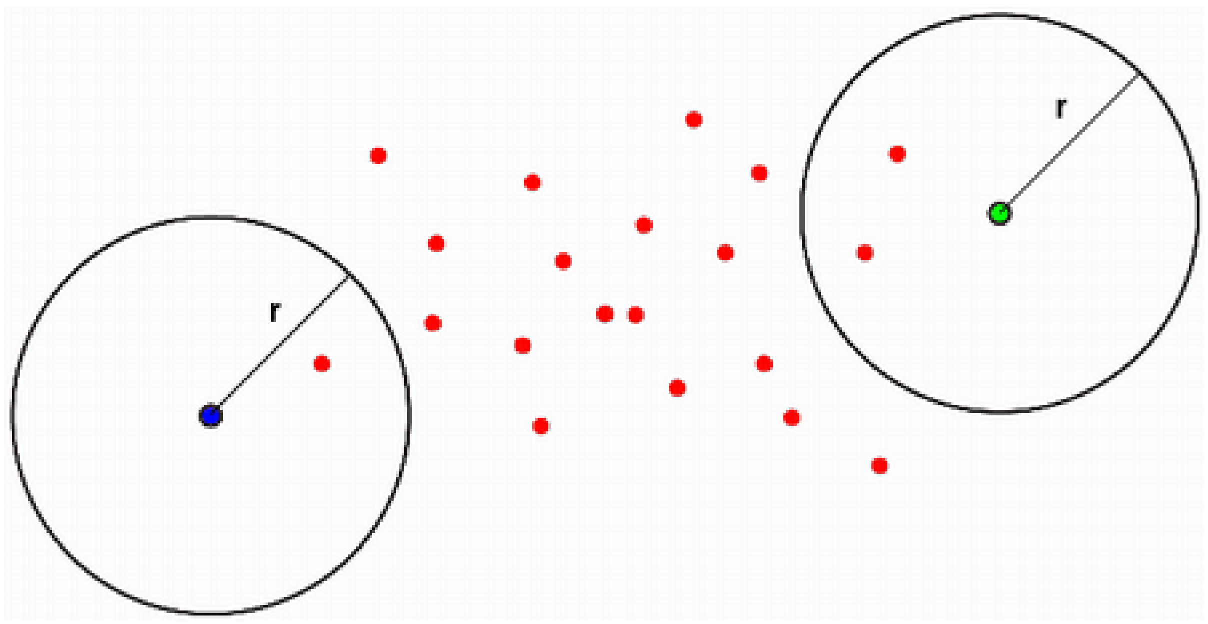


Figure 8: Illustration of the radius outlier filter. If the minimum number of neighbors is set to 2 and radius to r, the blue point will be removed and the green point will remain in the data set.

Finally an averaging is made with the data that is left after the filtering for the best TCP estimation.

Image Processing

The Kinect sensor provides the system with an RGB image, a depth image and a point cloud of the pallet containing the monitors. These images are used as input to the image processing unit. The main purpose of the image processing unit is to locate the monitors in the pallet. A simple flowchart of the input and output of the image processing unit is displayed in figure 9.

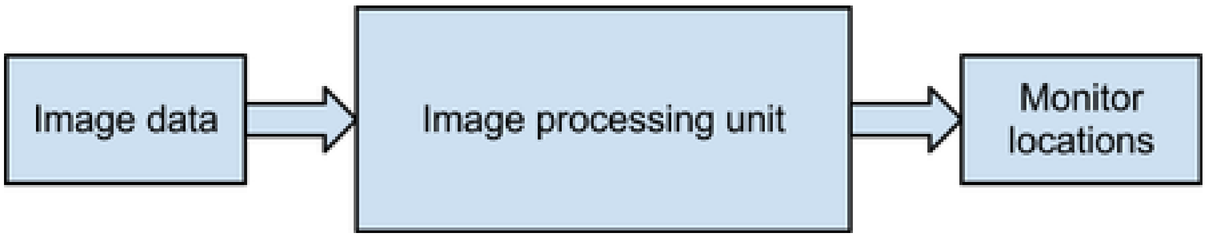


Figure 9: Flowchart of the image processing unit

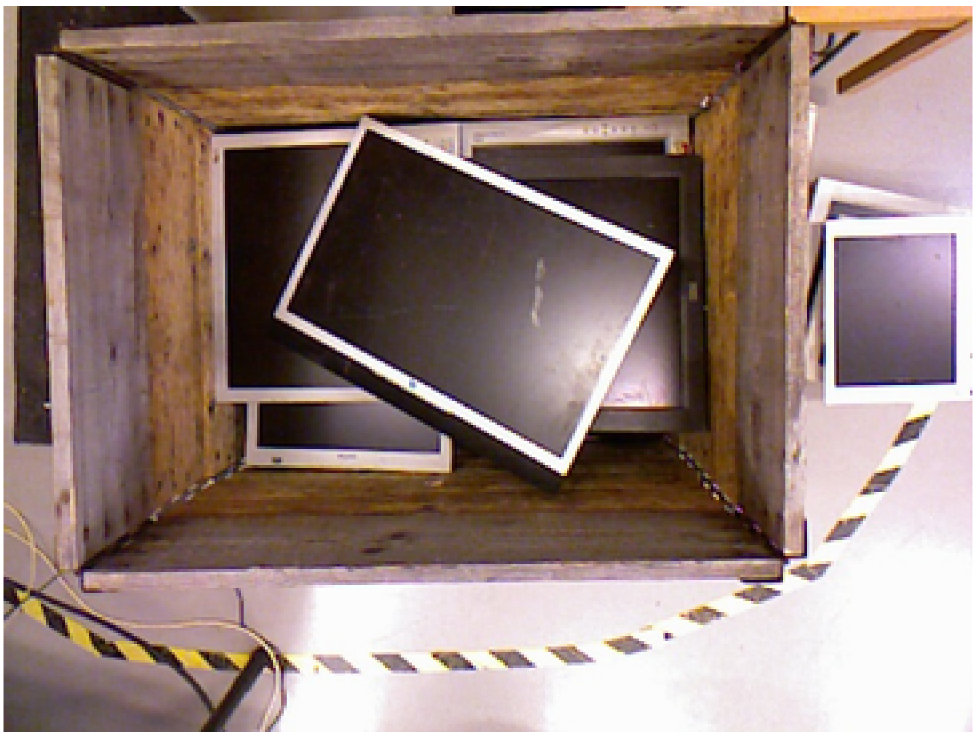


Figure 10: RGB image received from the Kinect sensor.

Plane segmentation

The first step is plane segmentation of the point clouds of the pallet received from the Kinect sensor. This was done using the Point Cloud Library[6] (PCL) which has plane segmentation functionality.

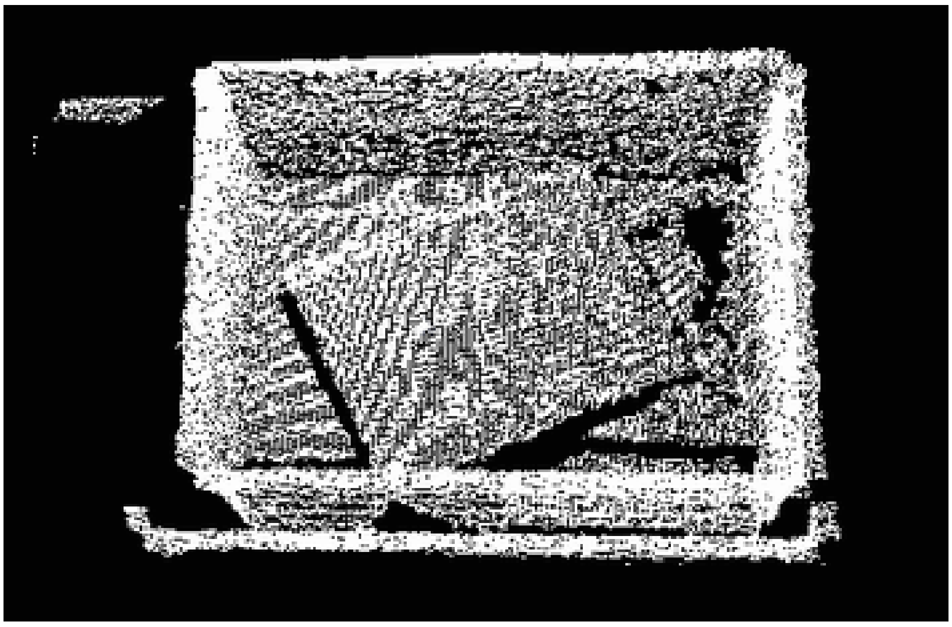


Figure 11: Point Cloud of the pallet received from the Kinect sensor.

In order to prevent the floor and the bottom of the pallet from being recognized as planar surfaces, all these points were removed from the point cloud before further processing. To remove the floor and the pallet bottom, a measurement was made to detect the distance between the Kinect camera and the bottom of the pallet. From this value, a threshold was found and all points below that value were labeled as floor or bottom-points and therefore removed from the point cloud.

To calculate the normals of each point in the point cloud, PCL's normal estimation using integral point cloud images with the covariance method is used. It creates 9 integral images to compute the normal for each point from the covariance matrix of its local neighborhood. The surface normal is the eigenvector corresponding to the smallest eigenvalue of the covariance matrix.

When the normal in each point in the point cloud has been estimated, planar regions are found by clustering these normal vectors. This is done based on the angle of the normal vectors as well as the distance between them. As the clustering is done, planar regions appear. Each planar region has a normal and a set of inliers.

This method returns all the planar regions in the scene, not only the monitors. Therefore, planar regions corresponding to the pallet have to be identified and removed from the monitor planes. The sides of the pallet are modeled as planes that are almost vertical, leading to a z-component that is almost zero in the plane's normal. This makes it easy to pick out all the vertical planes in the scene. That a plane is vertical does not automatically make it a part of the pallet. A monitor might be standing in a position giving it a vertical screen. This problem is taken care of by noticing that the pallet has a rectangular shape. Because of this, the normals of it's sides should be either orthogonal or parallel. By investigating the dot products between the normals, it is possible to determine which regions correspond to the pallet and which do not.

Outlier Removal

The plane model used for the segmentation doesn't have any additional restrictions other than that the whole surface should be connected. This leads to problem where two surfaces meet each other, for example where a monitor touches the box side. Since a plane model has an infinite extent, points belonging to the box side can incorrectly be classified as parts of the monitor. The bounding box of the monitor will therefore be too large and parts of the monitor will falsely be considered covered.

To solve this problem, all points are projected in each found plane. If the projected point is close enough (5 cm) to the original point, the plane is a match for the point. If a point is classified as belonging to more than one plane, it will simply be removed from the calculations.

Identification

As the image data have been segmented into planes, the pallet planes and incorrectly segmented points have been removed, the system has to decide which of the detected planes are monitor screens and which are not. This is decided by investigating if each plane meet a criteria that every plane corresponding to a monitor screen should meet. Since the range of monitor sizes is known, planes that are too small or too big can be excluded. Hence, the criteria is a size condition. The size can be determined by calculation the area. Planes that are too small or too big are ignored and the planes that remain are seen as monitor screens.

Monitor Selection

The system is designed to pick up one monitor at a time. Therefore, it is necessary to determine which monitor is the most suitable to pick up. There are several variables that affect the decision:

- A monitor that lies close to the top of the pallet should be easier to pick up than one further down in the pallet.
- A large monitor should be better to pick up than a small one since it frees more space in the pallet.
- A plane segment that is rectangular, i.e. looks like a standard monitor is less likely to be covered by a monitor than one that is not rectangular. Therefore, a rectangular plane segment should be more suitable to pick up.

To find the most suitable monitor to pick up, it is necessary to first decide the visibility and validity of the monitors and finally compare the monitors with respect to how high up in the pallet they are located.

In order to determine the visibility of each located monitor, a box is drawn using the maximum and minimum values in all directions of the points in the point cloud. The area of the drawn box is then compared to the area of the convex hull of the point cloud creating a percentage value where 100 % means that the monitor is entirely visible. Using a threshold of 80 % visibility, the monitors which are less suitable to pick up are filtered out.

As described in the identification section it is important to investigate that the monitors which passed the previous test is really a monitor and not just a small part of a monitor or a pallet wall, to do this check certain tests have to be passed in order som be labeled as a real monitor. These tests include:

- The area of the monitor's convex hull must exceed 700 square centimeters.
- The shortest side of the monitor's bounding box must exceed a certain value. In this case 20 cm was chosen since the diameter of the grasping tool is about 19 cm.
- The longest side of the monitor's bounding box must be below 60 cm. Otherwise the monitor is probably a side of the pallet.

When the unsuitable monitors have been filtered out and the validity of the monitors has been checked, the remaining monitors' values in the z-direction (upwards) are then compared to decide which monitor is located on top in the pallet.

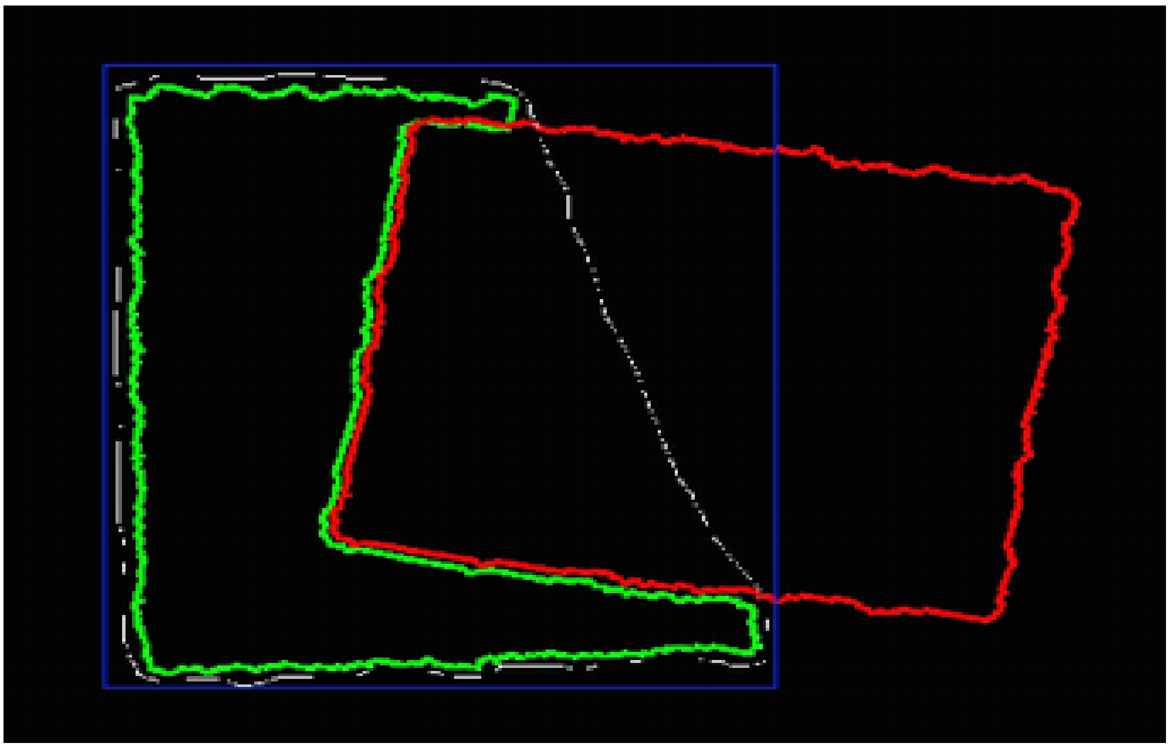


Figure 12: Image showing the monitors detected by the kinect. The red and green color represent monitors. The blue color represent the bounding box and the white color represent the convex hull.

In figure 12 above, the white area represents the convex hull of the point cloud and the blue area represents the box. It is notable that the green point cloud is less suitable to pick up than the red one, since the red monitor is almost 100 % visible according to the test explained above.

After each pick-up, a new image is evaluated in the same way to find the next suitable monitor to pick up.

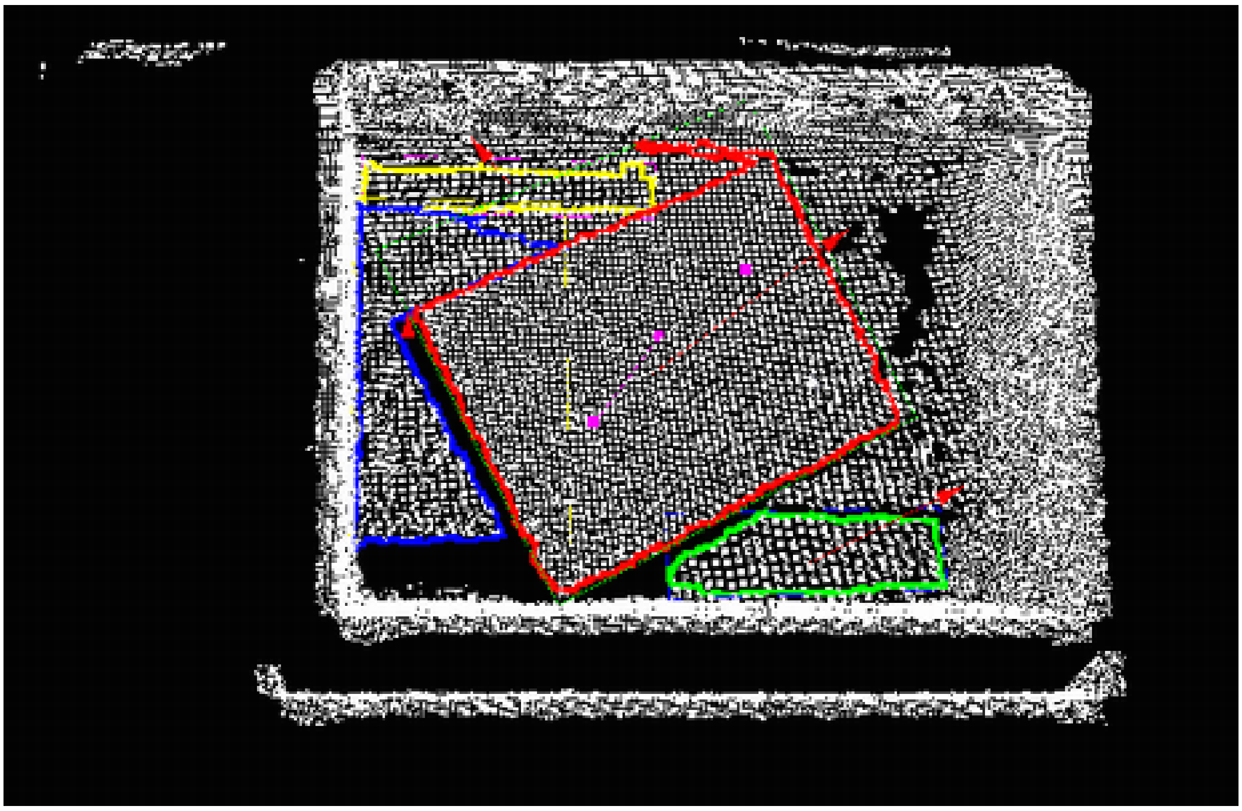


Figure 13: Point cloud of the pallet with all the regions found by the plane segmentation. The red region fulfills all the criteria for a monitor and is the most suitable to pick up. The yellow, green and blue regions are all monitors that isn't suitable for a pick-up. The purple "blobs" will be described in the planning chapter.

Planning

From the image processing unit, the monitor most suitable to pick up is given. To pick up a monitor, the robot needs instructions regarding where and in what direction to grasp. This is calculated by the planning unit. A simple flowchart of the input and output of the planning unit is displayed in the figure below.

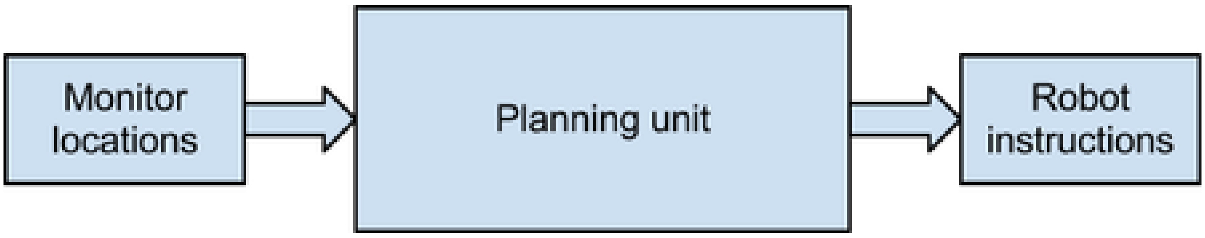
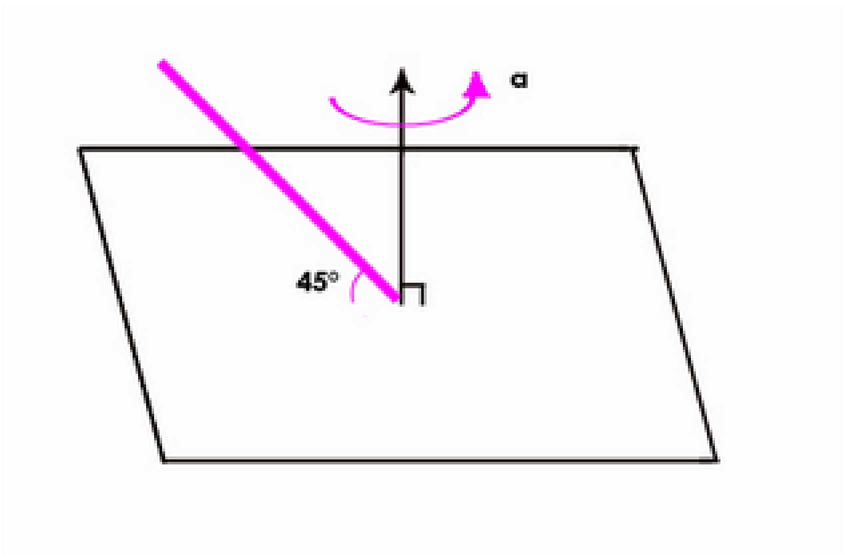


Figure 14: Flowchart of the planning unit

Calculation of Tool Pose

When it is determined which monitor is most suitable to pick up, the position and pose of the vacuum gripper has to be calculated. The position is represented in cartesian coordinates in the robot's global coordinate system and the pose by rotations around the X-, Y-, and Z-axis (Euler angles) in the same coordinate system.

The grasping point is simply selected as the center of the monitor screen's bounding box. The gripping tool is mounted on the robot arm at a 45° angle and the arm has to be rotated from the normal of the monitor with the same angle. The resulting angle of attack is therefore determined only up to a rotation around the monitor normal.



The last rotation is discretized and represented by 3D points where the robot arm can pass through, henceforth called *elbow points*. These points lie on a circle around the grasping point but are translated in the monitor normal direction to create the desired 45° angle to the monitor plane.

The physical limitations of the robot makes the choice of *elbow point* non-trivial:

1. There has to be a free space between the grasping point and the *elbow point*
2. The *elbow point* has to be in reach of the robot, with respect to the length of the robot arm and the restraints of all axes

To comply with point 1 the elbow point is chosen near the center of the box. That means that the last segment of the arm will be pointing towards a side of the box. There are however a few cases where this simple approach is not sufficient:

- If the monitor is right in front of the robot and has a normal point up and forward (away from the robot) the three outer segments of the arm align. This results is that the axes have to turn much faster than they can, and the robot's security mechanisms will stop it. To avoid it the arm has to be slightly bent at every angle.
- If the normal of the monitor points towards the middle of the box the *elbow point* closest to the middle will be too low. In this case another point has the be chosen.

When the angle of attack is determined, it has to be translated to three rotations around the robot's global cartesian axes. This is done by representing the wanted pose of the tool with three 3D points in the robot coordinate system and calculating the rigid transformation from the starting position of the tool: p1=(0, 0, 0), p2=(-25,0, 25), p3=(0, 0, 25). An illustration of this is found in figure 15 below.

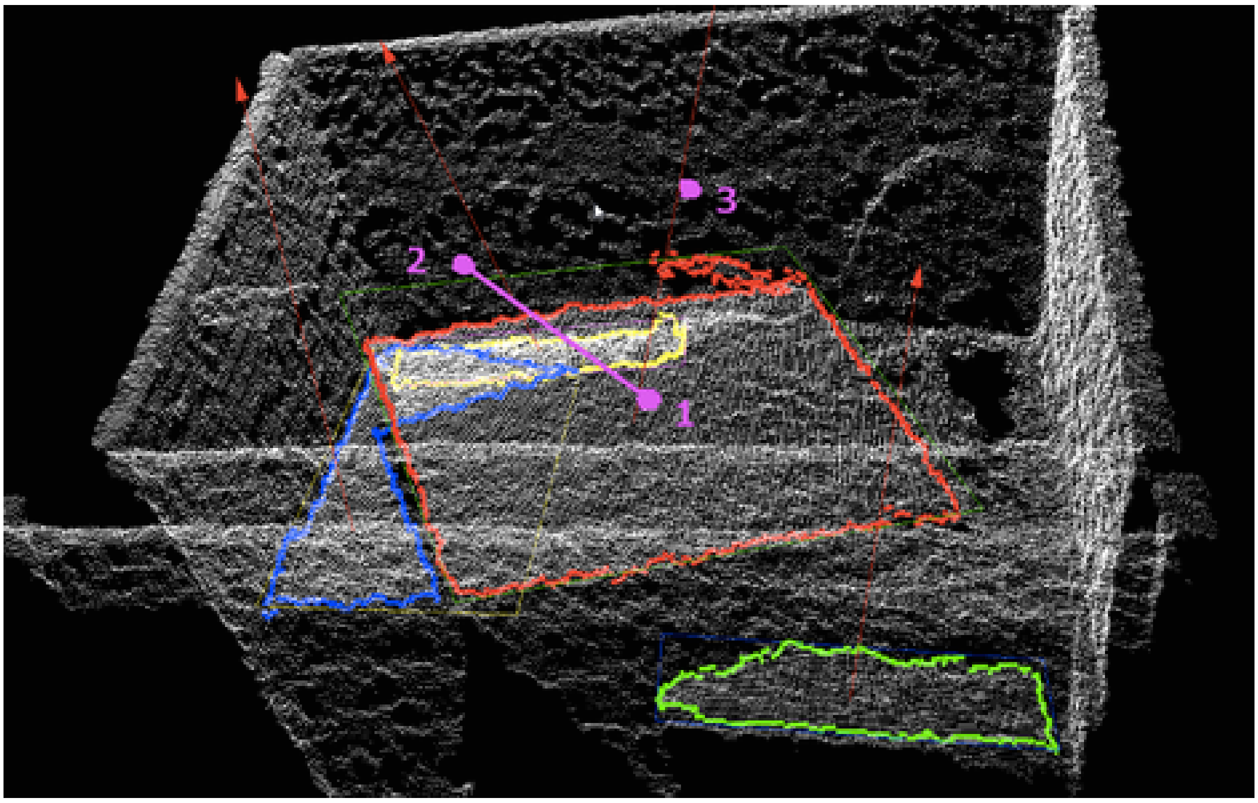


Figure 15: Point cloud of the pallet with the three points to calculate the robot arms rotation.

1: The grasping point in the center of the monitor's bounding box (TCP)
2: The elbow point. This point represents, together with the grasping point, the pose of the robot forearm along the magenta colored line.
3: Point 25 cm above the grasping point in the monitor screen's normal direction.

Path Planning

When it is determined which monitor is most suitable to pick up, a route from the robot arm's position to the grasping point of the monitor needs to be calculated. This route of a set of three sets of pose coordinates:

- A point in the center of the box 30 cm above the grasping point. The rotation of the tool is the wanted rotation for the pick-up. This ensures that the monitor will be lifted in a straight line in the center of the box.
- Tool placed a few centimeters above the monitor. The second pose coordinates in the set will be placed a few centimeters above the estimated grasping point of the

monitor. The robot arm will move linearly to this point in order to prevent collision with the pallet walls.

- The grasping point. To reach the grasping point of the monitor, the arm will move the last few centimeters linearly with the robot tool maintaining the previously calculated rotations.

The coordinates will place the robot arm in one of three different starting poses depending on the orientation of the monitor in the pallet. The different poses are calculated in order to avoid the axes of the robot arm from reaching its rotation limit, which will cause the system to stop.

Robot Communication

The Yaskawa Controller runs a web server and the communication is performed by http requests. The robot communication unit is implemented as a module that runs in parallel with the image processing and path planning algorithms. It connects to the Yaskawa Controller and establishes a session when the program is started. This session is maintained during the whole execution of the program. During the executing it keeps track of the state of the robot by constantly polling robot server.

The computer program initiates robot programs by setting variables. The robot triggers on the variables and starts the correct program.

Since the robot coordinate system differs from the Kinect coordinate system, all positions and angles have to be mapped to the robot coordinate system. This is done in the calibration unit by request of the robot communication unit.

Main Program

The main program connects all the different units and handles all transmission and reception between the robot and the program. Multiple threads are used to speed up the system. By doing so it is possible to start the image processing as soon as the robot arm is out of the Kinect's field of view. A flowchart of the main program is displayed in the figure below.

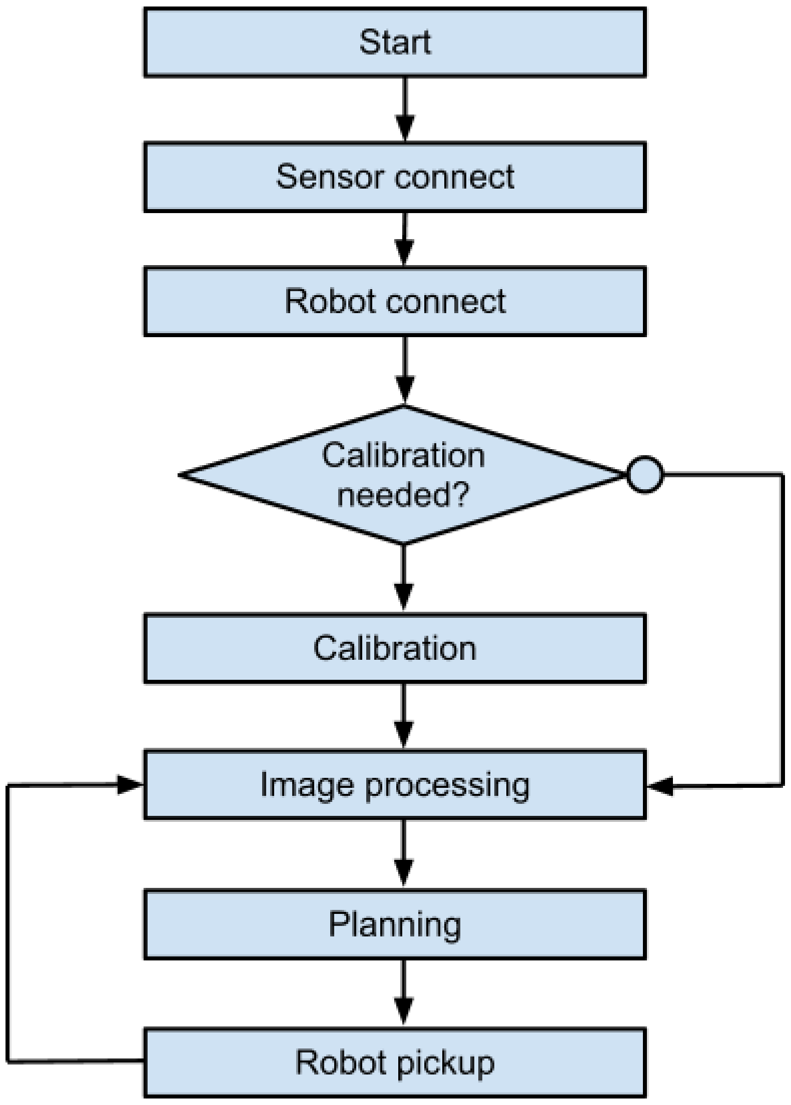


Figure 16: The structure of the main program

Graphical User Interface

The Graphical User Interface, GUI, is capable to start and stop the system. If the system encounters an error, such as failure to detect any monitors, information about it is displayed. For more information, see the user manual.

Robot Movement

The robot unit consists of Yaskawa NX100 Controller, a manual controller and Yaskawa SDA10 robot. The robot communication unit provides the robot with instructions. These instructions contain the set of coordinates that the robot arm, or more specifically the TCP, should move through to get to the grasping point of a desired LCD monitor. The robot arm has seven axis. Three cartesian coordinates and three rotations define a 3D point together with orientation. When adding a fourth rotation, the 7th axis of the robot, it is possible to define a unique pose of the robot.

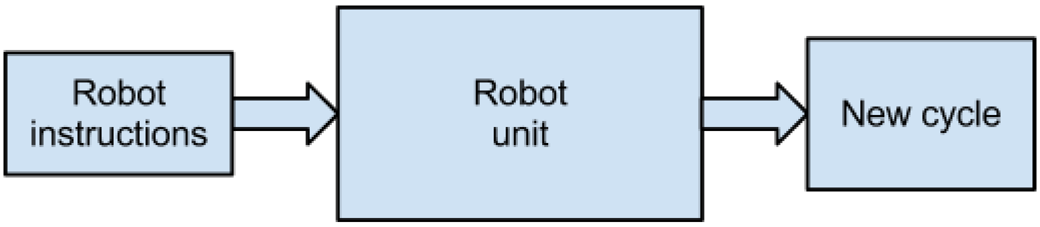


Figure 17: Flowchart over robot movement process.

Robot Programming

The manual controller is used to create a program which tells the robot to execute a certain instruction at a certain point in time. This program will control the robot's movements and actions. The structure of the robot program is described below.

Master Program

The Master Program moves the robot to a starting position, and then chooses a sub program depending on input from Robot communication unit.

Calibration Program

The Calibration Program moves the robot arm through a series of eight predetermined points. Upon reaching each point, the program waits for the kinect to take a snapshot and for that snapshot to be processed.

Pick Up Program

The Pick Up Program moves the robot through a set of predetermined pick-up points. The predetermined points include a starting point, a drop off point and a set of points which will force the arm to move along a certain path to avoid collision with the pallet and the kinect. The arbitrary points specify the robot's movements inside the pallet. How and why these points are calculated can be read in the *planning section.*

When the image processing is done and which monitor to pick up has been determined, the Image processing unit notifies the Robot unit. The robot arm is positioned above the pallet. The path from this point down to the grasping point of the monitor is decided by the pick-up points received from the Image processing unit. When the grasping tool has reached the grasping point, the robot arm will move backward through the pick-up points, in order to place the monitor at a user chosen drop off point. A flow chart of the pickup cycle can be found in figure 18 below.

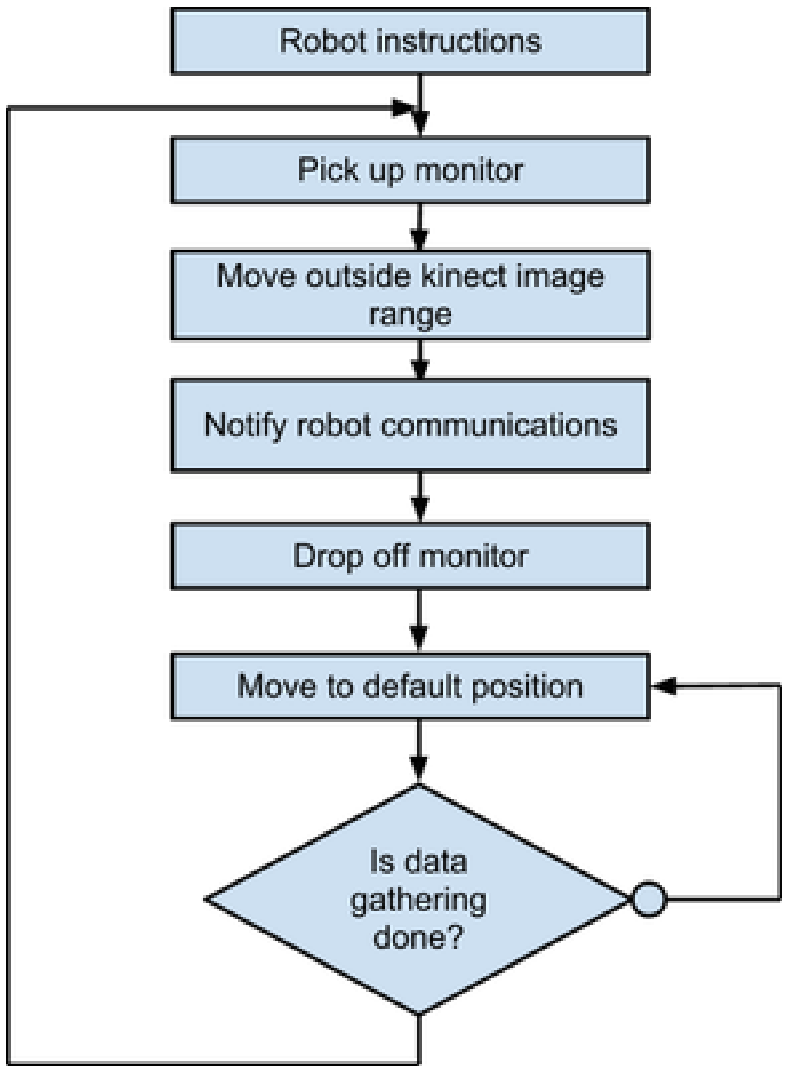


Figure 18: Flowchart of robot unit.

Result

In this section, some numerical values of the performance of the system is presented. The data was gathered from one set of monitors in a pallet and hence it does not represent the general performance of the system. The data has been divided into two sets depending on how many monitors were suited for pick-up. The segmentation algorithm for the case with one monitor suited for pick-up was able to find the monitor in four out of eight situations. In the case of two possible monitors, the segmentation algorithm detected 57 % of the monitors in 23 attempts. When a monitor had been selected, the system had a successful pick-up rate of 42 %. Most of the failures were caused by collision with the pallet walls but in a three out of 31 pick-ups the robot arm could not reach the desired grasping point.

Discussion

The main part of this section covers the possible future developments of the project. There are many things that could be done to improve the system. Some of these improvements are quite easy to implement and some are more complex.

Plane Segmentation

The plane segmentation algorithm is one of the weak components of the system. It does not always find all planar regions in the scene, which may cause the system to select to pick up a monitor that is lying partly underneath another monitor that would be a better option to pick up. Hence, to implement a more advanced plane segmentation algorithm would be a good way to improve the performance of the system. The group has found several interesting methods that are presented here.

SLIC Superpixel Segmentation

The SLIC superpixel segmentation algorithm[3] is an algorithm that uses the RGB image and groups adjacent pixels that are similar into larger segments called superpixels. The clustering is not made in the RGB space. The number of superpixels can be specified by the user. By selecting an appropriate number, a useful segmentation as in figure 19 can be obtained.

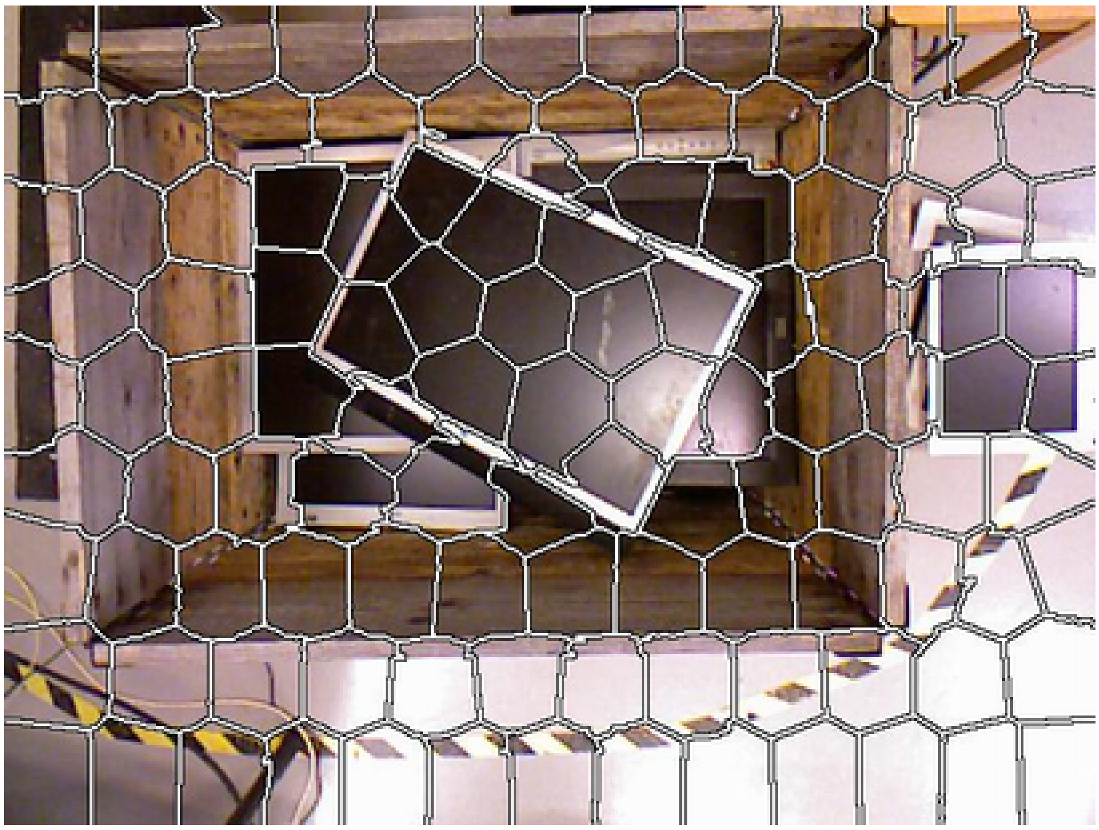


Figure 19: Result after the SLIC superpixel algorithm.

Using the three dimensional point cloud from the Kinect sensor, a normal for each superpixel can be estimated. Then, by clustering the normals, the superpixels corresponding to the parts of the same plane could be merged and the monitor planes in the scene detected.

Subspace Clustering

Subspace clustering is a method for finding lower-dimensional clusters embedded within higher-dimensional subspace clusters. For example, it can be used to find planes and lines in a three dimensional point cloud. One promising subspace clustering algorithm that the group looked into is the HiSC algorithm[4], which is a hierarchical subspace clustering algorithm.

Robustness of the System

To ensure that the system can perform a successful pickup with a higher probability, there are a few issues that could be solved if there were more time available.

Noisy Data

The three dimensional point cloud from the Kinect sensor are quite noisy and the plane segmentation has problems dealing with missing data. This is a problem, since it sometimes causes the plane segmentation to fail. If the noise prevent the algorithm to detect planar regions, no monitors can then be found by the system. There are some possible solutions to this problem. It could probably be solved by implementing more advanced plane segmentation algorithms. Another option is to apply some kind of smoothing on the Kinect data. One idea is to use a moving least squares algorithm for this, but since it is computationally very heavy and the amount of points in the Kinect point cloud is fairly large it leads to very slow data processing. To interpolate missing values, which probably would lead to better normal estimations, an idea is to use median filtering.

Path Planning

With the robot control system used in the project, the robot arm’s movement from one point to another is somewhat ambiguous. It is also not possible to read out of the control system what the robot's current pose and position is while it is in motion. The only things known are the initial and final pose of the tool. This pose does not define an unique configuration of the robot arm, since there are several configurations of the arm resulting in the same tool pose. This means that the robot arm’s movement between two coordinates can not be fully predicted. It may even be that the robot is incapable of moving from the initial pose to the final one, leading the robot arm into singular states which causes the entire system to fail. If the robot arm's current pose and position were always available, it would be possible to create a system that could predict if the movement from pose A to pose B is possible to execute. This would also make it possible to predict if the robot arm is not on a collision course with the pallet or other monitors.

By creating a system that is capable of predicting this type of problem the robustness of the system would increase significantly.

Speed of the System

It is possible to increase the speed of each pick up cycle by optimization of the system. By investigating and improving the path planning, and executing image processing and robot movement actions in parallel, the time of each pick up cycle can be reduced. The image processing software could also be rewritten to further improve performance.

The point cloud from the Kinect sensor is a large data set with $640*480=307200$ points. The large number of points makes processing slow. It should be possible to reduce the size of the data set and still get a good performance in the monitor detection. One way of reducing the number of points is to downsample the data using a 3D voxel grid. A voxel grid is a set of small three dimensional boxes in space over the input point cloud data. Then all points in each voxel are approximated by their centroid.

Flexibility of the System

To be able to handle more general scenarios, for example a case where some of the monitors are placed with the screen facing downward it is necessary to use additional tools. The second robot arm can be equipped with a gripping claw in order to pick up the monitors with screens facing downward. This would make the entire recycling process much more efficient and flexible. The process of detecting monitors in the scene would also have to be improved. Since the monitors lying upside down in general are not flat, the idea of looking for planar regions must be modified. Instead of modeling the monitors just as planes, a more advanced model is needed. It would also be necessary to use other and better segmentation methods.

References

The references used in this document are listed below.

[1]Linköping University, ISY, (2012), *Project Course TSBB11*
http://www.cvl.isy.liu.se/education/undergraduate/tsbb11

[2]Willow Garage et al. (2009), *Open Source Natural Interaction*
http://openni.org/

[3]Achanta, R, et al. (2012), *SLIC Superpixel Compared to State-of-the-Art Superpixel Methods*
http://ivrg.epfl.ch/supplementary_material/RK_SLICSuperpixels/index.html

[4]Achtert, E, et al. (2006), *Finding Hierarchies of Subspace Clusters*
Institute for Informatics, Ludwig-Maximilians-Universität München, Germany

[5] Itseez, et al. (2012), *Open Source Computer Vision*
http://opencv.org/

[6](2012), *Point Cloud Library*
http://pointclouds.org/

[7]Duda, R.O. and Hart, P.E., et al. (1972), *Use of the Hough Transformation to Detect Lines and Curves in Pictures*
Stanford Research Institute, Menlo Park, California

[8]Canny, J, et al. (1986), *A Computational Approach to Edge Detection*
http://perso.limsi.fr/Individu/vezien/PAPIERS_ACS/canny1986.pdf