Scaled Demo of Autonomous Load Handling between Ship and Shore

Martin Mæland, Tarjei Skotterud, Martin Dahlseng Hermansen

***Abstract*—To automate load handling in a real-life and full-scale scenario the technology must be tested and verified over a long time frame in both simulation and in the real world. In this paper we present an approach for a down-scaled demo of autonomous load handling between ship and shore. To do that we present methods to; navigate and drive a mobile robot in an imitated port area using NAV2, detect and estimate the pose of a simulated load using machine vision based on YOLOv7, simulate a ship by applying an ocean wave spectra to a desktop-sized Stewart platform, and control of robotic arms with motion compensation located in dynamic environments. This resulted in two different demonstrations. The first is displaying a scaled demo of autonomous load handling on shore performed by a mobile robot. While the latter consists of performing a pick and place procedure between ship and shore in a simulated environment, whilst compensating for the simulated wave motion. This suggests that an approach for a scaled demo of autonomous load handling between ship and shore is provided.**

***Index Terms*—Computer vision, machine learning, robotics, autonomous load handling, ROS 2**

I. Introduction

AUTONOMOUS load handling between ship and shore is a challenging task many of the largest organizations in the offshore industry are working towards. By automating this process the operation can yield in more reliable and repeatable performance that saves time and resources by offering more consistent results. The automotive industry has come far in delivering autonomous cars, and the mechanical industry has used autonomous robots in lifting operations for several decades. However, combining those technologies to work seamless and safe in a dynamic environment that is constantly changing is a tedious task that may take years, if not decades to perfect.

Some ports across the world have autonomous terminals up and running today, however they are only used for specific kinds of loads and tasks. Even though their work field is narrow they are seen as the beginning of the future. The shipping industry is changing rapidly with more advanced ships coming every year. They are becoming greener with the use of more sustainable fuel, and more reasonable in use by being able to cut staff because of newer technology. The ships are also able to travel at a higher speed, and hence using less time traveling between destinations. This has led to bottlenecks being caused in many of the larger ports in the world causing delay, and hence, waste of money and resources. The next step for the industry is to look forward and automate the process that can be seen as the congestion; the operation of unloading and loading the ships faster to reduce downtime.

To fully automate load handling in a real life scenario the technology must be investigated, tested and verified over a longer time frame to determine its safety and reliability. Most of the technologies to perform the given task of autonomous load handling in cargo ports are already made and widely investigated.

In this paper we present how different technologies can be combined to create an approach for autonomous load handling between ship and shore. Methods are introduced for how each component in the system interacts with each other to perform the given task. This is done in order to have a working demonstration that later can be used for continued research in the field.

A demonstration of autonomous load handling between ship and shore is dependent on numerous of factors. To create the base layer for further development it is chosen to scale the demonstration for the sake of simplicity, and to neglect aspects concerning corporate social responsibilities.

II. Related work

The academic research for the task of autonomous load handling between ship and shore is slender. Approaches for object detection have seen an immensely increase of performance since convolutional neural network entered the field with 'Region-Based Convolutional Neural Network' (R-CNN) [1] and 'You Only Look Once' (YOLO) [2]. Since then numerous new versions has improved the detection accuracy and processing time, with one of the latest being YOLOv7 [3].

Literature on object pose estimation is broad. There are several different approaches for addressing the problems of pose estimation, with the more recent approaches focusing on deep learning [4] [5]. However, by using depth images from stereo camera the pose of given objects can as well be accurately estimated by running algorithms on the data to detect planes and other relevant information needed in order to estimate the pose [6].

For robot localization, navigation, and path planning ROS Navigation stack has been one of the most popular solution for over a decade. In the recent years it has however seen a downfall due to lack of keeping up with modern trends. That changed in 2020 when a research group proposed NAV2 for ROS2 [7]. NAV2 builds on the legacy of ROS Navigation, but have an increased capacity in dynamic environments because it is applicable to a wider variety of modern sensors.

Both pick-and-place and motion compensation are frequently researched topics. Large factories have become heavily

Martin Mæland, Tarjei Skotterud, Martin Dahlseng Hermansen is with the Department of Engineering Sciences, University of Agder, Grimstad, Norway

dependent on robotic arms to perform repetitive tasks, to the extent that researchers are now looking into ways to reduce the energy consumption of such operations [8] [9]. Additionally, research is conducted to find ways to pick up objects with variable shapes to increase the area of use [10] [11].

In the research community, motion compensation has been in focus for several years, and advanced compensation algorithms based on deep learning [12] and system modeling [13] has been developed. A common application for motion compensation is load handling in dynamic environments such as offshore. Efforts have been made to improve both ship-to-shore and ship-to-ship, with inverse kinematic control [14] and anti-swing assistant [15].

III. Method

A. Experimental setup

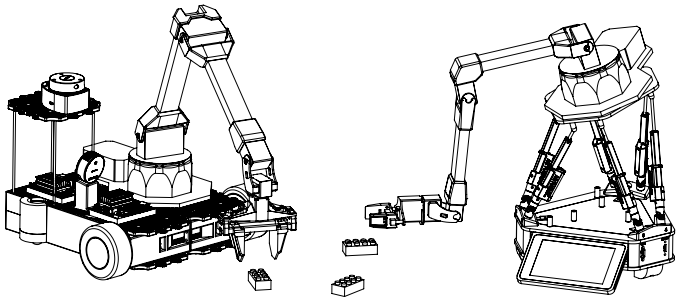


Fig. 1. Experimental setup used for the scaled demo of autonomous load handling between ship and shore.

The experimental setup for the scaled demo of autonomous load handling between ship and shore consists of three main subsystems; the load, an autonomous vehicle and a simulated ship. To simulate the load we have used three different kinds of 2x4 Lego Duplo bricks in the colors red, green and blue.

To imitate an arbitrary industrial truck the UiAbot is used as the autonomous vehicle to handle the load. The UiAbot is a differential driven robot that can either be autonomously driven with NAV2, or manually controlled with a keyboard/controller [16]. It is also equipped with an Intel RealSense L515 camera, and a ViperX-300 5DOF robotic arm from Trossen Robotics. The camera is used to stream RGB and aligned depth frames with ROS2 to other applications in order to gather information about the load. Whereas the robotic arm is used to pick and place the load.

In order to simulate a ship that is floating in the ocean we are using a desktop-sized Stewart platform made at UiA [17] to simulate the waves, and a WidowX-250 6DOF robotic arm from Trossen Robotics to pick and place the load between shore and ship.

Every component used in the scaled demo is connected to the same network, and communicates using ROS 2 as shown in Fig. 2.

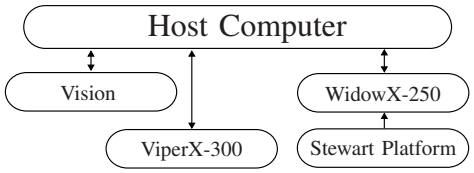


Fig. 2. Communication flow.

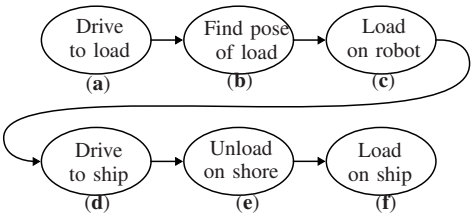


Fig. 3. Flow diagram of system operations: (**a**) (**d**) Navigation and localization with UiAbot [16], (**b**) Object Detection & Pose Estimation, (**c**) (**e**) (**f**) Pick and Place.

B. Object Detection & Pose Estimation

To detect the bricks and estimate their pose a camera mounted on the UiAbot is streaming a RGB stream, and a aligned depth stream using the ROS 2 Wrapper for Intel RealSense Devices. Those two streams are used to detect the detectable objects, and estimate the pose of the objects using Algorithms 1 and 2.

Algorithm 1 Object Detection

Input: RGB image from camera, I_{rgb}
Output: Bounding boxes of detected objects, B_{rgb}

while videostream **do**
  $I \leftarrow I_{rbg}(n_h \times n_w = 640 \times 640)$ ▷ pad image to match a stride $= 32$.
  $B \leftarrow$ inference ▷ detect trainable objects in I.
  $B_{nms} \leftarrow \varnothing$ ▷ initialize empty set.
  for $b_i \in B$ **do**
    $discard \leftarrow False$ ▷ keep or discard.
    for $b_j \in B$ **do**
      if $same(b_i, b_j) > \lambda_{nms}$ **then**
        if $score(c, b_j) > score(c, b_i)$ **then**
          $discard \leftarrow True$
        end if
      end if
    end for
    if not $discard$ **then**
      $B_{nms} \leftarrow B_{nms} \cup b_i$ ▷ add b_i to list.
    end if
  end for
  $B_{rgb} \leftarrow \varnothing$ ▷ initialize empty set.
  for $b_i \in B_{nms}$ **do**
    $b \leftarrow$ scale b_i to I_{rgb}
    $B_{rgb} \leftarrow B_{rgb} \cup b$ ▷ add b to list.
  end for
end while

Algorithm 2 Pose Estimation

Input: RGB image from camera, I_{rgb}; aligned depth image from camera, D; bounding box from inferencing, B; Camera intrinsic matrix, K; Transformation matrix from map to camera, H_c^m

Output: Pose of brick, H_b^m; color of brick, c

$I_B \leftarrow I_{rgb}[B]$
$I_{c_1}, I_{c_2} \leftarrow K - means(I_B, k = 2)$ ▷ K-means algorithm to segment brick and background in I_B.
$I_{seg} \leftarrow I_{c_1} \wedge I_{c_2}$ ▷ largest set from clustering is the load.
$c \leftarrow max(I_{seg}[r], I_{seg}[g], I_{seg}[b])$ ▷ estimate color on brick by find the color channel with highest value.
$M \leftarrow I_{seg}(\text{where } I_{seg} > 0)$ ▷ convert segmented image to image mask.
$D_M \leftarrow D$ & M ▷ mask aligned depth image with logical and operation.
$\boldsymbol{P}_c \leftarrow K^{-1} \cdot D_M$ ▷ get pointcloud of brick.
$\boldsymbol{S}_1, \boldsymbol{S}_2 \leftarrow RANSAC(\boldsymbol{P}_c)$ ▷ determine two planes in $\boldsymbol{P}_c$.
$P_{11}, P_{12} \leftarrow$ extreme points in xy for $\boldsymbol{S}_1$
$P_{21}, P_{22} \leftarrow$ extreme points in xy for $\boldsymbol{S}_2$
$d_1, d_2 \leftarrow$ distance between P_{n1} and P_{n2}
$d_l \leftarrow max(d_1, d_2)$ ▷ length of brick long side.
$P_{l1}, P_{l2} \leftarrow$ points used to find d_l
$\{b\} \leftarrow max(p_{l1,y}, p_{l2,y})$ ▷ brick origin is to the left on bricks long side.
$\boldsymbol{t} \leftarrow \{b\}$
$\boldsymbol{R} \leftarrow Kabsch(\{\vec{b}\}_x, \{\vec{c}\}_x)$ ▷ Kabsch algorithm to find rotation between $\{b\}$ and $\{c\}$ [18].
$\boldsymbol{H}_b^c \leftarrow \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}$
$\boldsymbol{H}_b^m \leftarrow \boldsymbol{H}_c^m \cdot \boldsymbol{H}_b^c$
return $\boldsymbol{H}_b^m$, c

Since the camera is moving along with the mobile robot it is necessary to determine the bricks position and orientation relative to map ({m}) instead of camera frame as seen in Fig. 4. The transformation matrices $\boldsymbol{H}_b^c$ and $\boldsymbol{H}_c^m$ is respectively found from the pose estimation and from the localization of the UiAbot. Using these transformation matrices the transformation matrix $\boldsymbol{H}_b^m$ can be found and used to place a brick relative to the world coordinate system.

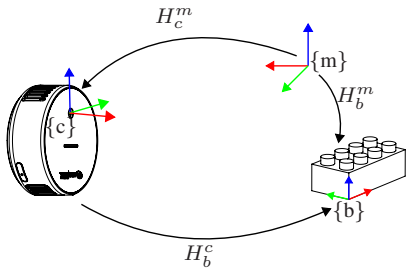


Fig. 4. Determining a bricks position and orientation relative to map coordinate frame. Note: We have rotated the camera coordinate system.

The dataflow from the camera stream to how the bricks are detected, and how their pose is estimated can be seen in Fig. 5.

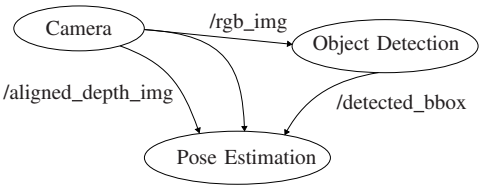


Fig. 5. The dataflow of the vision part from how the bricks are detected to how their pose is estimated.

C. Stewart Platform

1) Inverse kinematics: The inverse kinematics of the general Stewart platform corresponds to the prismatic joint displacement for each of the six links and their time derivatives based on given cartesian pose and velocities of the tool relative to base. This is usually done in two stages, where the inverse pose solution is used to find the geometrical jacobian in order to compute joint velocities.

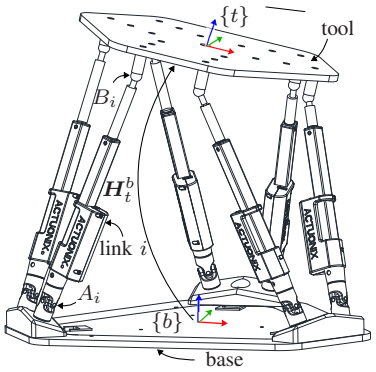


Fig. 6. Stewart platform used for this experiment.

Starting with the inverse pose problem, there are two coordinate frames attached to the robot. One in the stationary base and one on the moving tool. The transformation between these two frames is related with the homogeneous transformation matrix $\boldsymbol{H}_t^b$ as seen in Fig. 6. Furthermore, the lower and upper attachment points of the *i*-th link is described with A_i and B_i respectively.

$$\boldsymbol{p}_{B_i}^b = \boldsymbol{p}_t^b + \boldsymbol{R}_t^b \boldsymbol{p}_{B_i}^t \tag{1}$$

With Equation (1) we can describe the position of B_i with reference to $\{b\}$, where $\boldsymbol{p}_t^b$ and $\boldsymbol{R}_t^b$ is the position and orientation extracted from the tool transformation matrix $\boldsymbol{H}_t^b$ respectively. Once it is established, the i-th link vector is expressed by subtracting the position vector of A_i.

$$\boldsymbol{p}_{B_i/A_i}^b = \boldsymbol{p}_{B_i}^b - \boldsymbol{p}_{A_i}^b \tag{2}$$

The link length, denoted as L_i, is then described by the euclidean norm of Equation (2).

$$L_i = ||\boldsymbol{p}_{B_i/A_i}^b||_2 \tag{3}$$

$$\hat{\boldsymbol{u}}_i = \frac{\boldsymbol{p}^b_{B_i/A_i}}{L_i} \tag{4}$$

The pointing direction of each link expressed in $\{b\}$ is specified with the unit vector as shown in Equation (4). Ultimately, the effective displacement of the prismatic joints ($q_{1..6}$) is found by subtracting the minimum length of the link, e.g. L_i when $q_i = 0$.

$$q_i = L_i - L_{min} \tag{5}$$

Moving on to the inverse rate kinematics we obtain the velocity of point B_i with respect to $\{b\}$ by differentiating Equation (1) with respect to time. The given 3-vector of tool velocities with respect to $\{b\}$, is denoted as $\boldsymbol{\omega}^b_t$.

$$\dot{\boldsymbol{p}}^b_{B_i} = \dot{\boldsymbol{p}}^b_t + \boldsymbol{\omega}^b_t \times \boldsymbol{R}^b_t \boldsymbol{p}^t_{B_i} \tag{6}$$

The velocity of the prismatic joints ($\dot{q}_{1..6}$) is then calculated as the dot product of Equation (6) with the unit vector from Equation (4).

$$\dot{q}_i = \dot{\boldsymbol{p}}^b_t \cdot \hat{\boldsymbol{u}}_i + \boldsymbol{\omega}^b_t \times \boldsymbol{R}^b_t \boldsymbol{p}^t_{B_i} \cdot \hat{\boldsymbol{u}}_i \tag{7}$$

Merging the equations together for each of the six legs we get the 6×6 inverse jacobian matrix. This relates the desired tool velocity with the necessary equivalent in each joint, seen in Equation (8).

$$\boldsymbol{J}_{inv} = \begin{bmatrix} \hat{\boldsymbol{u}}_1^T & (\boldsymbol{R}^b_t \boldsymbol{p}^t_{B_1} \times \hat{\boldsymbol{u}}_1)^T \\ \vdots & \vdots \\ \hat{\boldsymbol{u}}_6^T & (\boldsymbol{R}^b_t \boldsymbol{p}^t_{B_6} \times \hat{\boldsymbol{u}}_6)^T \end{bmatrix} \tag{8}$$

Multiplying the inverse jacobian with a vector containing the desired 6 DOF velocity of $\{t\}$ with respect to $\{b\}$, outputs the six joint velocities in vector form.

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{inv} \begin{bmatrix} \dot{\boldsymbol{p}}^b_t \\ \boldsymbol{\omega}^b_t \end{bmatrix} \tag{9}$$

2) Forward kinematics: The forward pose problem is a much harder problem to solve due to the multiple number of possible joint configurations for a given tool pose. The forward pose problem of a Stewart platform with general geometry can have up to 40 solutions.

In order to find the pose ($\boldsymbol{\eta}$) of $\{t\}$ with respect to $\{b\}$, an iterative kinematic solver has to be implemented. It consists of an error function (f_e) with the latest pose estimate ($\hat{\boldsymbol{\eta}}$) as input.

$$f(\hat{\boldsymbol{\eta}}) = \boldsymbol{q} - \boldsymbol{q}(\hat{\boldsymbol{\eta}}) \tag{10}$$

Where $\boldsymbol{q}$ is the 6-vector of current joint position and $\boldsymbol{q}(\hat{\boldsymbol{\eta}})$ is the inverse position function that calculates the joint positions necessary to produce current pose estimate. The iterative solvers job is essentially to estimate poses until the error is within a defined accuracy threshold.

The chosen minimize method is a modified version of the *Levenberg-Marquardt* algorithm (LNA), also known as the damped least-squares method [19]. It relies on the jacobian matrix which has to be computed every iteration for the latest estimate. Solving this problem is done by first calculating the matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{J}^T_{inv} \boldsymbol{J}_{inv} + diag(\boldsymbol{J}^T_{inv} \boldsymbol{J}_{inv})\lambda \tag{11}$$

Where λ is the damping factor.

$$\boldsymbol{\delta} = \boldsymbol{A}^{-1} \boldsymbol{J}_{inv}(\boldsymbol{q} - \boldsymbol{q}(\hat{\boldsymbol{\eta}}_{k-1})) \tag{12}$$

$$\hat{\boldsymbol{\eta}}_k = \hat{\boldsymbol{\eta}}_{k-1} + \boldsymbol{\delta} \tag{13}$$

Then the least-squares problem is solved using Equation (12) and, if acceptable, appended to the previous pose estimate by Equation (13). These three equations are put inside a loop and iteratively solved until an estimate is within the defined accuracy threshold or the number of iterations has reached its set maximum. The implemented algorithm is described in greater detail in Algorithm 3.

Algorithm 3 Stewart platform forward kinematics using LMA

Input: Joint position vector, $\boldsymbol{q}$; Initial configuration pose guess, $\hat{\boldsymbol{\eta}}_0$; Accuracy threshold, ϵ; Maximum number of iterations for the estimation loop k_{max}; Maximum number of iterations for the damping factor adjustment loop, i_{max};

Output: Final configuration pose $\boldsymbol{\eta}$

$\hat{\boldsymbol{\eta}} \leftarrow \hat{\boldsymbol{\eta}}_0$, $\lambda \leftarrow \lambda_0$, $k \leftarrow 0$
while $(notfound)$ and $(k < k_{max})$ **do**
  $k \leftarrow k + 1$
  $err \leftarrow ||\boldsymbol{q} - \boldsymbol{q}(\hat{\boldsymbol{\eta}})||_2$ ▷ error between actual and estimated joint positions.
  if $err < \epsilon$ **then**
    $found \leftarrow True$ ▷ solution within tolerance.
  else
    $\boldsymbol{J}_{inv} \leftarrow \boldsymbol{J}_{inv}(\hat{\boldsymbol{\eta}})$ ▷ inverse jacobian based on latest pose estimate.
    for $i = 0$ to i_{max} **do**
      $\boldsymbol{A} \leftarrow \boldsymbol{J}^T_{inv} \boldsymbol{J}_{inv} + diag(\boldsymbol{J}^T_{inv} \boldsymbol{J}_{inv})\lambda$
      $\boldsymbol{\delta} \leftarrow \boldsymbol{A}^{-1} \boldsymbol{J}^T_{inv} err$ ▷ solve least squares.
      $\hat{\boldsymbol{\eta}}^* \leftarrow \hat{\boldsymbol{\eta}} + \boldsymbol{\delta}$ ▷ set intermediate pose estimate.
      if $||\boldsymbol{q} - \boldsymbol{q}(\hat{\boldsymbol{\eta}}^*)||_2 < err$ **then**
        $\hat{\boldsymbol{\eta}} \leftarrow \hat{\boldsymbol{\eta}}^*$ ▷ step accepted.
        $\lambda \leftarrow \lambda / \lambda_{down}$ ▷ adjust damping factor downwards.
        break for-loop
      else
        $\lambda \leftarrow \lambda \cdot \lambda_{up}$ ▷ step rejected, adjust damping factor upwards.
      end if
    end for
  end if
end while
return Final configuration pose $\boldsymbol{\eta} \leftarrow \hat{\boldsymbol{\eta}}$.

Even though there are simpler methods of solving non-linear least squares problems, such as the *Gauss-Newton* algorithm or the method of *gradient descent*, the LNA was preferred mainly for its robustness when given an inaccurate initial guess.

The LNA can be thought of as a hybrid method that interpolates between the two others by adjusting the damping parameter λ. Choosing this parameter, as well as its adjustment gains, often requires some trial and error. For the final algorithm we settled with an initial damping of 1.0, with adjustment factors of 1.5 and 5.0 for λ_{up} and λ_{down} respectively.

The choice of accuracy threshold and maximum number of iterations depends on how much time is allocated between each execution of the forward kinematics algorithm, as well as how accurate the solution has to be. In order to fit the rate of the control loop (100 Hz), the accuracy threshold ϵ is set to $0.1mm$, with k_{max} and i_{max} equal to 20 and 5 respectively.

$$\begin{bmatrix} \dot{\boldsymbol{p}}_t^b \\ \boldsymbol{\omega}_t^b \end{bmatrix} = \boldsymbol{J}_{inv}^{-1} \dot{\boldsymbol{q}} \tag{14}$$

Now that the pose part of the forward kinematics is established, we can find the decomposed velocities of $\{t\}$ with respect to $\{b\}$ by turning around Equation (9), as shown in Equation (14).

D. Robotic Arm

1) Forward Kinematics: The forward kinematics of the robotic arm is solved to find the gripper position $\boldsymbol{p}_g^b$ and orientation $\boldsymbol{R}_g^b$ relative to the base frame, given joint angles $\boldsymbol{q}$. A common method to solve the forward kinematics problem is to use the Denavit-Hartenberg convention, a systematic approach to define the rigid motion between joint q_{i-1} and q_i with homogeneous transformation matrix $\boldsymbol{H}_i^{i-1}$, expressed in Equation (15) [20].

$$\boldsymbol{H}_i^{i-1} = \boldsymbol{T_z}(d_i)\boldsymbol{R_z}(\theta_i)\boldsymbol{T_x}(a_i)\boldsymbol{R_x}(\alpha_i) \tag{15}$$

Each transformation is given by translating along the z-axis, rotating about the z-axis, translating along the x-axis, and rotating about the x-axis. Defined by the four DH parameters θ_i, d_i, a_i and α_i, respectively. The translation and rotation matrices are:

$$\boldsymbol{T_z}(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{R_z}(\theta_i) = 1 \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{T_x}(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{R_x}(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The DH parameters for the WidowX-250 6DOF robotic arm are given in Table I, and shown in Fig. 7 [21]. In this case, the base frame is considered equal to frame $\{0\}$ of joint q_1, and will hereafter be referred to interchangeably.

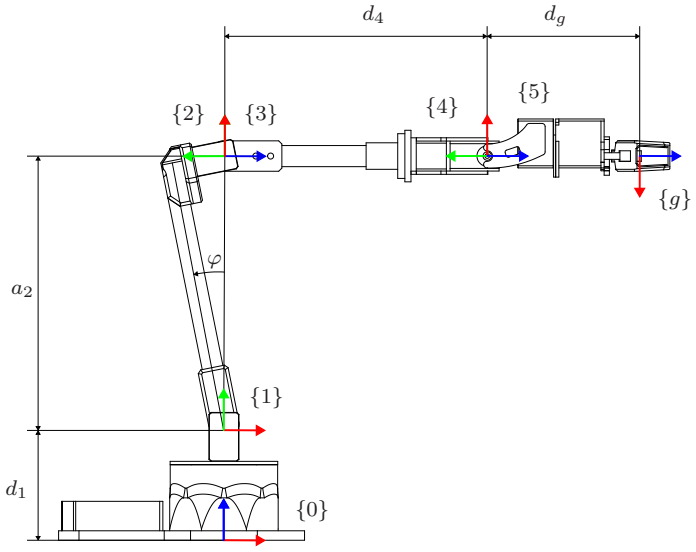


Fig. 7. Side view of robotic arm, annotated with frames and dh parameters.

TABLE I
DH TABLE FOR THE WIDOWX-250 6DOF ROBOTIC ARM.

Link i	d_i	θ_i	a_i	α_i
1	d_1	q_1	0	$\frac{\pi}{2}$
2	0	$q_2 - \varphi + \frac{\pi}{2}$	a_2	0
3	0	$q_3 + \varphi$	0	$\frac{\pi}{2}$
4	d_4	q_4	0	$-\frac{\pi}{2}$
5	0	q_5	0	$\frac{\pi}{2}$
6	d_g	$q_6 + \pi$	0	0

Furthermore, we find the transformation from $\{0\}$ to $\{g\}$ by calculating the product of the homogeneous transformations between each joint, where $\{g\}$ equals $\{6\}$:

$$\boldsymbol{H}_g^0(\boldsymbol{q}) = \boldsymbol{H}_6^0(\boldsymbol{q}) = \prod_{i=1}^{6} \boldsymbol{H}_i^{i-1} \tag{16}$$

The gripper position $\boldsymbol{p}_g^0$, and orientation $\boldsymbol{R}_g^0$ is then found in $\boldsymbol{H}_g^0$.

2) Jacobian: The Jacobian matrix gives us the relationship between gripper task space velocities $\boldsymbol{\nu}_g^0$, and joint space velocities $\dot{\boldsymbol{q}}$:

$$\boldsymbol{\nu}_g^0 = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{17}$$

Given a robot configuration, we are able to find each joint velocity contribution to the gripper velocity, which makes up the columns in the Jacobian. Joint q_i will provide an angular velocity expressed as:

$$\boldsymbol{\omega}_g^i = \left(\boldsymbol{R}_g^i \hat{\boldsymbol{u}}_z\right) q_i \tag{18}$$

And provide a linear velocity expressed as:

$$\boldsymbol{v}_g^i = \left(\boldsymbol{R}_g^i \hat{\boldsymbol{u}}_z \times \boldsymbol{p}_g^i\right) q_i \tag{19}$$

The Jacobian can then be written as:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{R}_g^0 \hat{\boldsymbol{u}}_z \times \boldsymbol{p}_g^1 & \dots & \boldsymbol{R}_g^i \hat{\boldsymbol{u}}_z \times \boldsymbol{p}_g^i \\ \boldsymbol{R}_g^0 \hat{\boldsymbol{u}}_z & \dots & \boldsymbol{R}_g^i \hat{\boldsymbol{u}}_z \end{bmatrix} \tag{20}$$

Since the WidowX-250 6DOF robotic arm has six joint velocities, and the task space velocity of the gripper is given by three angular and three linear velocities, the Jacobian matrix is a square $6x6$ matrix. As long as the matrix determinant is non-zero, Equation (17) can be inverted, and we can calculate the joint velocities given gripper task space velocity as:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\boldsymbol{\nu}_g^0 \tag{21}$$

Robotic arms with more or less than six joints will have a non-square and non-invertible Jacobian matrix. In addition, when facing a singularity, that is when then determinant is zero, the square matrix is also non-invertible. A more general expression for the joint velocities $\dot{\boldsymbol{q}}$ using the pseudoinverse $\boldsymbol{J}^{\dagger}$ can be written as [22]:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\boldsymbol{\nu}_g^0, \text{ where } \boldsymbol{J}^{\dagger} = (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T \tag{22}$$

3) Inverse Kinematics: The inverse kinematics problem is solved to find joint angles $\boldsymbol{q}$ based on the gripper position $\boldsymbol{p}_g^0$ and orientation $\boldsymbol{R}_g^0$, making up the robot pose $\boldsymbol{\eta}_g^0$. This problem is often harder than the forward kinematics problem, due to the fact that there may be several solutions.

For an articulated robotic arm with a spherical wrist, the solution of the problem can be found using Piepers method. His approach splits the problem into two parts, by first determining q_1, q_2, and q_3 based on the wrist position $\boldsymbol{p}_w^0$. Then determining q_4, q_5, and q_6 based on the gripper orientation $\boldsymbol{R}_g^0$. The wrist position $\boldsymbol{p}_w^0$ is found by translating the distance d_g along the negative z-axis of $\{g\}$, and is equal to $\boldsymbol{p}_4^0$ (Equation (23)). [23]

$$\boldsymbol{p}_w^0 = \boldsymbol{p}_g^0 - \boldsymbol{R}_g^0\hat{\boldsymbol{u}}_z d_g \tag{23}$$

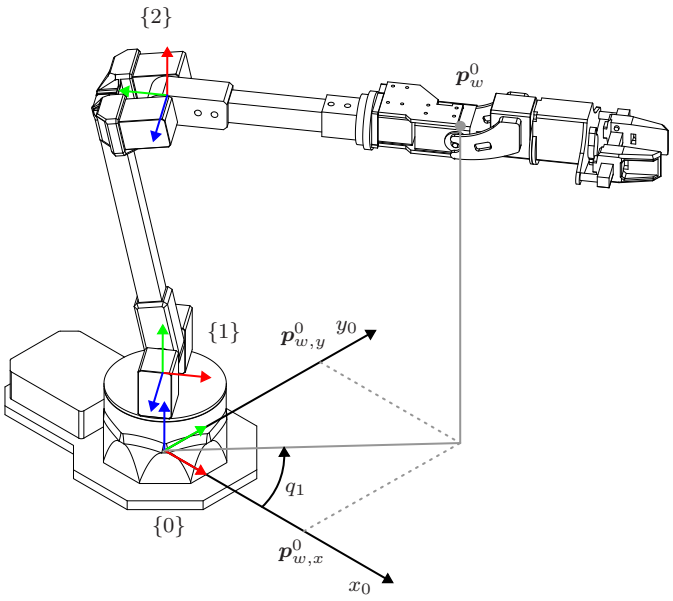


Fig. 8. Finding the first joint angle based on wrist position.

Based on Equation (23) we can determine q_1, since q_2 and q_3 does not contribute to a rotation about the z-axis of $\{1\}$ (Equation (24)).

$$q_1 = atan2(\boldsymbol{p}_{w,y}^0, \boldsymbol{p}_{w,x}^0) \tag{24}$$

Furthermore, q_2 and q_3 will only contribute to a change in position in the x- and y-axis of $\{1\}$. Therefore, if we transform to $\{1\}$ to solve for q_2 and q_3, the problem can be reduced to a simple geometrical problem (Fig. 9).

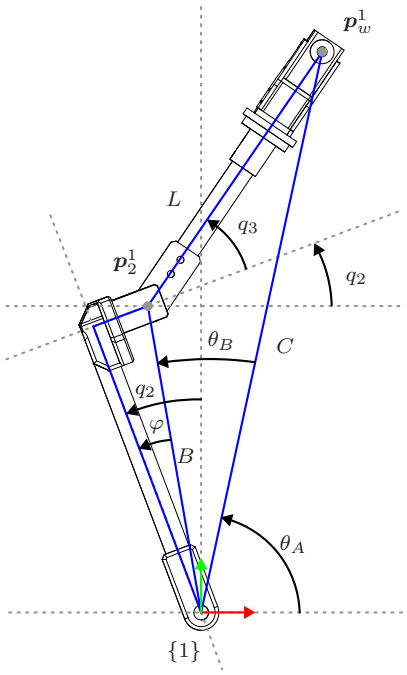


Fig. 9. Finding the second and third joint angles by using geometric relations.

To determine q_2, we first need to find θ_A and θ_B. θ_A can be expressed as:

$$\theta_A = atan2(\boldsymbol{p}_{w,y}^1, \boldsymbol{p}_{w,x}^1) \tag{25}$$

To find θ_B, we use the law of cosines on the triangle enclosed by the lines B, C, and L (Equation (26)). Which can also be written as $||\boldsymbol{p}_2^1||_2$, $||\boldsymbol{p}_w^1||_2$, and $||\boldsymbol{p}_{w/2}^1||_2$, respectively.

$$L^2 = B^2 + C^2 - 2BCcos(\theta_B) \tag{26}$$

$$\Rightarrow \theta_B = \pm acos\left(\frac{B^2 + C^2 - L^2}{2BC}\right) \tag{27}$$

Note that in Equation (27), there exist a positive and a negative solution. These represent the elbow up and elbow down configurations, respectively. We have used the elbow up configuration for the WidowX-250 6DOF robotic arm. Furthermore, q_2 can be determined as:

$$q_2 = \theta_A + \theta_B + \varphi - \frac{\pi}{2} \tag{28}$$

The third angle q_3 is then given by:

$$q_3 = atan2(\boldsymbol{p}_{w/2,y}^1, \boldsymbol{p}_{w/2,x}^1) - q_2 \tag{29}$$

Continuing, q_4, q_5, and q_6 can be determined by inspecting $\boldsymbol{R}_g^3$ and comparing it to the symbolic expression of $\boldsymbol{R}_g^3$

(Equation (32)). $\boldsymbol{H}_g^3$ can be found by premultiplying the transformation matrix from base to gripper with the transformation matrix from base to joint 4:

$$\boldsymbol{H}_g^3(q_1, q_2, q_3) = (\boldsymbol{H}_3^0)^{-1}\boldsymbol{H}_g^0 l \tag{30}$$

$$\boldsymbol{R}_g^3(q_4, q_5, q_6) = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{21} & e_{32} & e_{33} \end{bmatrix} \tag{31}$$

$$= \begin{bmatrix} s_4 s_6 - c_4 c_5 c_6 & s_6 c_4 c5 + s_4 c_6 & c_4 s_5 \\ -s_6 c_3 - s_4 c_5 c_6 & s_4 s_6 c_5 - c_4 c_6 & s_4 s_5 \\ s_5 c_6 & s_5 s_6 & c_5 \end{bmatrix} \tag{32}$$

In which s_i represents $sin(q_i)$, and c_i represents $cos(q_i)$. Subsequently, the e_{ij} entry in $\boldsymbol{R}_g^3$ must match the equivalent entry in the symbolic expression. Starting with e_{33}, we are able to solve for q_5 in Equation (33). Note that there is a positive and a negative solution, which corresponds to the wrist up or wrist down configuration, respectively.

$$q_5 = \pm cos^{-1}(e_{33}) \tag{33}$$

Depending on the chosen configuration for q_5, we can solve for the last two joints. If the configuration is wrist up, the following is valid for q_4 and q_6:

$$q_4 = atan2(e_{23}, e_{13}) - \pi \tag{34}$$

$$q_6 = atan2(-e_{32}, e_{31}) - \pi \tag{35}$$

If the configuration is wrist down, the following is valid for q_4 and q_6:

$$q_4 = atan2(e_{23}, e_{13}) \tag{36}$$

$$q_6 = atan2(-e_{32}, e_{31}) \tag{37}$$

4) 5DOF Robotic Arm: Solving the kinematics of the 5DOF robotic arm is similar to the solution of the 6DOF robotic arm. With the main difference being that the 5DOF robotic arm has one less joint in the wrist, resulting in one less degree of freedom in the toolpoint orientation. The DH parameters for the ViperX-300 5DOF robotic arm are given in Table II [24].

TABLE II
DH TABLE FOR THE VIPERX-300 5DOF ROBOTIC ARM.

Link i	d_i	θ_i	a_i	α_i
1	d_1	q_1	0	$\frac{\pi}{2}$
2	0	$q_2 - \varphi + \frac{\pi}{2}$	a_2	0
3	0	$q_3 + \varphi - \frac{\pi}{2}$	d_4	0
4	0	$q_4 + \frac{\pi}{2}$	0	$\frac{\pi}{2}$
5	d_g	$q_5 + \pi$	0	0

E. Motion

1) Ocean-Wave Spectra: The Stewart platform, which is used to simulate the deck of a ship, has sine-wave generators built in for each degree of freedom. For a more realistic motion, an enhancement to the onboard software was made in order to generate and execute stochastic wave motion based on an Ocean-Wave Spectra. The wave spectra most relevant for ships are the two-parameter *Pierson-Moskowitz* (PM) and *Jonswap* wave spectra. The equations for the two wave spectras was implemented in accordance with DNVGL-CG-0130 (p. 21-23) [25], where the PM wave spectrum for a fully developed sea is given by Equation (38).

$$S_{PM}(\omega) = \frac{5}{16} \cdot H_s^2 \omega_p^4 \cdot \omega^{-5} exp\left(-\frac{5}{4}\left(\frac{\omega}{\omega_p}\right)^{-4}\right) \tag{38}$$

Where H_s is the significant wave height in meters, ω is the frequency in rad/s and ω_p is the spectral peak frequency calculated from the wave mean period T_p as $w_p = 2\pi/T_p$ in rad/s.

$$S_J(\omega) = A_\gamma S_{PM}(\omega)\gamma^{exp\left(-0.5\left(\frac{\omega-\omega_p}{\sigma\omega_p}\right)^2\right)} \tag{39}$$

The Jonswap wave spectrum is formulated as a modification of the PM wave spectrum, where it is multiplied with an extra peak enhancement factor γ^r to represent a sea state in a fetch limited situation. It is described by Equation (39), where:

$\gamma = 3.3$ is a non-dimensional peak shape parameter
expected to be a reasonable model for $3.6 < T_p/\sqrt{H_s} < 5$.

$\sigma =$ spectral width parameter
0.07 for $\omega \leq \omega_p$
0.09 for $\omega > \omega_p$

$A_\gamma = 1 - 0.287\ln(\gamma)$ is a normalizing factor

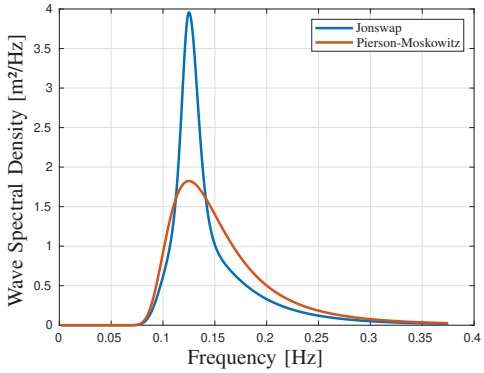


Fig. 10. Plot of Jonswap and PM wave spectral density with $H_s = 4m$ and $T_p = 8s$.

A plot of both wave spectras are shown in Fig. 10 with a significant wave height (H_s) of $4m$ and a frequency range of $[0.01..3\omega_p]$, where T_p is set to 8 seconds.

With the spectra calculated for N number of frequencies inside the frequency-range, the amplitude of each wave component is calculated using Equation (40), where $\Delta\omega$ is the

considered frequency interval. The phase (ϕ_i) for each wave component is then sampled from the uniform distribution defined with range $[0..2\pi]$.

$$A_i = \sqrt{2S(\omega_i)\Delta\omega_i} \quad \text{for } i = 1..N \tag{40}$$

Furthermore, considering that a simulation of a specific ships dynamics in the described waves was not of interest in this experiment, a "raft in slow waves" assumption is made when calculating the six degree of freedom body displacement's amplitudes and phase lags. This assumption essentially means that the ship will follow the wave surface like a raft, where an amplitude gain (A_k) and a phase lag (ϕ_k) is calculated for each degree of freedom depending on the encountered wave angle θ, seen in Fig. 11 [26]. At last, the tool frame displacement in each degree of freedom ($\delta_{1..6}$) is calculated by the sum of all wave components using Equation (41).

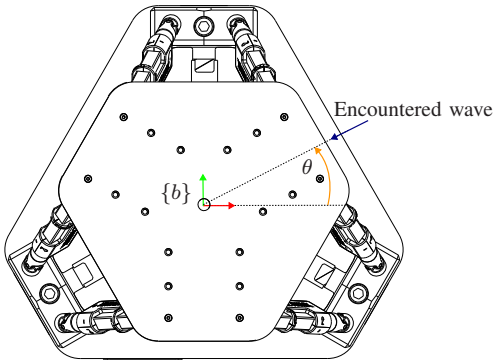


Fig. 11. Stewart platform from above showing the encountered wave angle given in the fixed base frame..

$$\delta_k = \sum_{i=1}^{N} A_i A_k cos(\omega_i t + \phi_i + \phi_k) \quad \text{for } k = 1..6 \tag{41}$$

The wave generated and used in this experiment is based on the Jonswap spectrum shown in Fig. 10 with an encountered wave angle (θ) of 25°. Additionally, in order to comply with the Stewart platform motion range, the amplitudes in xyz is scaled down by a factor of 10^{-2}. It must be stated that this is not a typical, or in any way realistic ship motion in dock, but chosen to display an interesting demo as well as the capabilities of the compensation control system.

2) Compensation: A closed loop joint space control system is implemented to regulate each joint velocity based on the desired gripper task space pose $\boldsymbol{\eta}^n_{g,d}$ and joint position feedback $\boldsymbol{q}$ (Fig. 12). In which the frame $\{n\}$ is the static frame of the Stewart platform tool in neutral. The desired task space position is generated by using third order polynomials for each coordinate axis, and the desired task space orientation is generated from the spherical linear interpolation (SLERP) [27]. Combined, this leads to a smooth Cartesian motion. Furthermore, the desired gripper pose is transformed to desired joint space position $\boldsymbol{q}_d$ using the inverse kinematics (III-D3).

In order to pick up a stationary load relative to the wave motion of the robot base, the robot must compensate in all six degrees of freedom. Using only a position controller would result in unsatisfactory results, as the controller would only output a signal to the joints when there exist an error. This is not ideal for systems with non-zero accelerations. Therefore, the Stewart tool task space velocities $\boldsymbol{\nu}^n_t$ are transformed to task space velocities in the robot base frame $\{0\}$. Then converted to joint velocities $\dot{\boldsymbol{q}}_t$ by using the pseudo-inverse of the Jacobian, and subtracted from the joint velocity references from the controller $\dot{\boldsymbol{q}}_d$ to get the final velocity signals to the robot joints $\dot{\boldsymbol{q}}_{d,t}$.

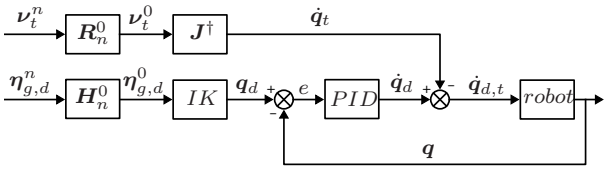


Fig. 12. Control system for WidowX250 6DOF robotic arm.

3) Pick and Place: Pick and place is implemented by introducing a sequence of actions, starting with calculating the desired gripper pose $\boldsymbol{\eta}^0_{g,des}$ relative to the given brick pose $\boldsymbol{\eta}^0_b$. This relation enables us to adjust how we want to grasp the brick with the static transform $\boldsymbol{H}^b_{g,des}$. For the 5DOF robotic arm we decided to grasp the brick from above with the z-axis of the gripper facing downwards along the bricks negative z-axis. Mainly because of the grippers limited orientation. As a result the 5DOF robotic arm is able to pick up the brick from any pose on the ground, as long as it is within reach. The 6DOF robotic arm, on the other hand, is set to grasp the brick from the short hand side with the gripper z-axis along the bricks x-axis.

After the desired gripper pose is calculated, we generate a task space trajectory by interpolating between the current gripper position $\boldsymbol{p}^0_g$ and desired gripper position $\boldsymbol{p}^0_{g,des}$ with a third order polynomial for each axis. Unlike the position, the interpolation between the current gripper orientation $\boldsymbol{R}^0_g$ and the desired gripper orientation $\boldsymbol{R}^0_{g,des}$ is implemented differently for the 5DOF robot and 6DOF robot because of the limitations of the 5DOF robotic arm. Where the gripper is set to always face downwards. Leading to one degree of freedom left to actuate, rotation about the gripper z-axis. The only joint contributing to this rotation is q_5. From the given brick pose we can find the desired rotation of this joint, and interpolate by using a third order polynomial. The trajectory for the 6DOF robotic arm orientation is discussed in III-E2.

By running the trajectories we get task space references which are transformed to joint space references with the inverse kinematics. For the 5DOF robotic arm these joint space references are sent directly to the robotic arm which is running in position mode. Unlike the 6DOF robotic arm, where the references are sent to the control system discussed in III-E2.

Furthermore, after completing the motion, the gripper is closed and the same procedure is used to get back to the initial pose. The actions performed to pick up the brick, is similar to the actions performed to place the brick, with the main difference that the gripper is opened when the gripper is at the desired pose.

IV. Results

A. Object Detection

For training the object detection model a dataset of 6030 images is created (Fig. 13), where 78% of them is used for training, 20% for validation, and 2% for testing.

Fig. 13. Example of one of the generated images in the dataset.

The object detection model is trained for 400 epochs with a batch size of 20. That resulted in a object detection model having an AP of about 0.95 and a recall of about 0.98 measured in the validation dataset (Fig. 14).

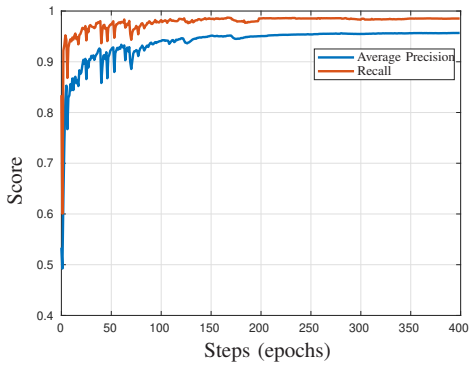
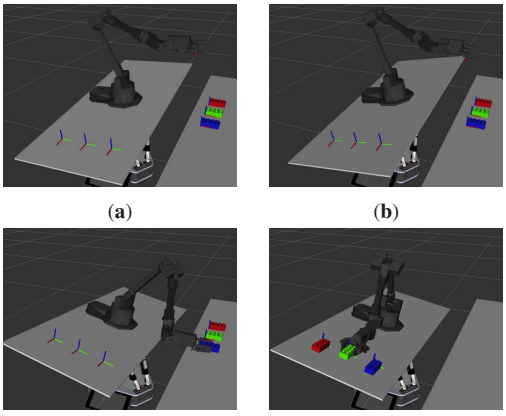


Fig. 14. Learning curve for average precision and recall.

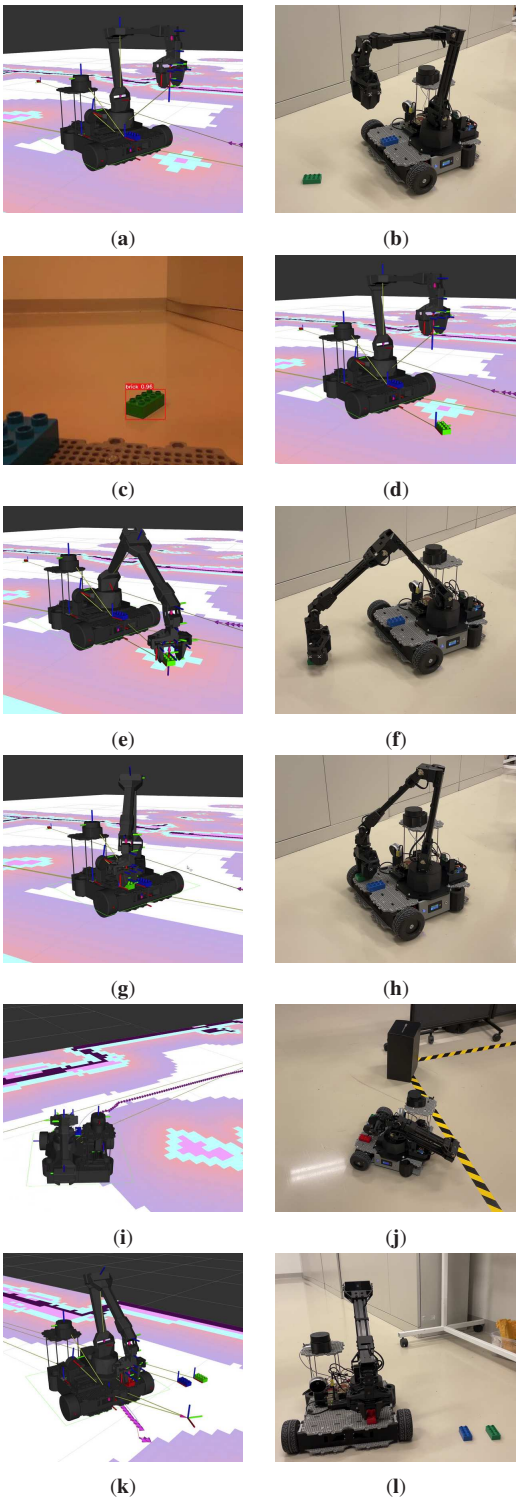
B. Motion Compensation



(a) (b) (c) (d)

Fig. 15. Image sequence of a pick and place procedure of the simulated ship visualized in RViz: (**a**) Robotic arm not compensating for the wave motion generated by the Stewart platform. (**b**) Robotic arm compensating for the wave motion generated by the Stewart platform. (**c**) Robotic arm compensating for the subjected wave motion and picking up the blue brick from shore. (**d**) Robotic arm not compensating for the wave motion to place the green brick on ship.

C. Pick & Place



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Fig. 16. Image sequence of a demo for the pick and place procedure on the UiAbot with real-time visualization in RViz: (**a**) (**b**) Initial position of UiAbot in RViz and from camera. (**c**) Inferencing detecting green brick. (**d**) Estimated pose on green brick successfully, green brick is shown in RViz. (**e**) (**f**) Robotic arm picking up green brick. (**g**) (**h**) Robotic arm placing green brick on UiAbot. (**i**) UiAbot driving and detecting obstacle. (**j**) UiAbot successfully avoiding the obstacle. (**k**) (**l**) UiAbot reached the final destination and placing the bricks on the shore for the simulated ship to pick them up.

V. Conclusion and Discussion

In this paper we have presented a scaled demo of autonomous load handling between ship and shore, and methods for how the technologies used in the overall system can be implemented. That is accomplished by performing demonstrations of a pick and place procedure on the mobile robot, and a pick and place procedure on the simulated ship. To navigate and drive around in the prior demonstration a UiAbot equipped with NAV2 is used. In order to detect the simulated load of Lego Duplo bricks a YOLOv7 object detection model is trained and used. The position and pose of the simulated load is estimated by creating a pointcloud using depth images from a Intel RealSense L515 camera. The ship motion is simulated with a Jonswap wave spectrum on the Stewart platform, where its forward kinematics based on actuator feedback is used as a reference for the robot arm compensation system.

Both the forward and inverse kinematics have been derived for the robotic arms in order to transform between task and joint space. The ViperX-300 5DOF robotic arm is mounted on the UiAbot and used for pick and place on shore, by sending position references directly to each joint. While the WidowX-250 6DOF robotic arm is mounted on the Stewart platform to perform pick and place during compensation of the simulated ship motion. However, because of instability in the built-in motor controllers on the 6DOF robotic arm, and considering that improving the controllers is beyond the scope of the experiment, we decided to use the robotic arm in a simulated environment with the proposed control system (Fig. 12).

In conclusions, by successfully demonstrating a pick and place procedure on a scaled system, we have provided an approach for autonomous load handling between ship and shore.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[3] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022. [Online]. Available: https://arxiv.org/abs/2207.02696

[4] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," 2018. [Online]. Available: https://arxiv.org/abs/1809.10790

[5] H. Chen, P. Wang, F. Wang, W. Tian, L. Xiong, and H. Li, "Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation," 2022. [Online]. Available: https://arxiv.org/abs/2203.13254

[6] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 536–551.

[7] S. Macenski, F. Martin, R. White, and J. G. Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2020. [Online]. Available: https://doi.org/10.1109%2Firos45743.2020.9341207

[8] M. Pellicciari, G. Berselli, F. Leali, and A. Vergnano, "A method for reducing the energy consumption of pick-and-place industrial robots," *Mechatronics*, vol. 23, no. 3, pp. 326–334, 2013.

[9] R. Balderas Hill, S. Briot, A. Chriette, and P. Martinet, "Performing energy-efficient pick-and-place motions for high-speed robots by using variable stiffness springs," *Journal of Mechanisms and Robotics*, vol. 14, no. 5, p. 051004, 2022.

[10] B. Imtiaz, Y. Qiao, and B. Lee, "Implementation of reinforcement learning algorithms for robotic pick and place with non-visual sensing," 2022.

[11] Q. Chen, L. Wan, P. Ravichadran, Y.-J. Pan, and Y. K. Chang, "Vision-based impedance control of a 7-dof robotic manipulator for pick-and-place tasks in grasping fruits," 2022.

[12] X. Guo, X. Zhang, X. Tian, W. Lu, and X. Li, "Probabilistic prediction of the heave motions of a semi-submersible by a deep learning model," *Ocean Engineering*, vol. 247, p. 110578, 2022.

[13] B. Rossin, A. Sreenivasan, and B. De Kruif, "Coupled control for motion compensated offshore operations," in *SNAME 27th Offshore Symposium*. OnePetro, 2022.

[14] S. S. Tørdal, G. Hovland, and I. Tyapin, "Efficient implementation of inverse kinematics on a 6-dof industrial robot using conformal geometric algebra," *Advances in Applied Clifford Algebras*, vol. 27, no. 3, pp. 2067–2082, 2017.

[15] S. S. Tørdal, "Real-time motion compensation in ship-to-ship load handling," 2019.

[16] "UiAbot," https://uiabot.dundermifflin.no/, accessed: 2022-11-10.

[17] T. Skotterud, J. Dale, K. Sand, and G. Myrland, "Development of a portable stewart platform for demonstrations and educational purposes," *B.Sc. thesis, University of Agder*, 2021, Link to report.

[18] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, Sep 1976. [Online]. Available: https://doi.org/10.1107/S0567739476001873

[19] R. Fletcher, "A modified marquardt subroutine for non-linear least squares," 1971.

[20] R. S. Hartenberg and J. Denavit, *Kinematic synthesis of linkages*. New York, NY: McGraw-Hill, 1964.

[21] "WidowX-250 6DOF — interbotix x-series arms documentation," https://www.trossenrobotics.com/docs/interbotix_xsarms/specifications/wx250s.html, accessed: 2022-11-10.

[22] M. B. Kjelland, I. Tyapin, G. Hovland, and M. R. Hansen, "Tool-point control for a redundant heave compensated hydraulic manipulator," *IFAC Proceedings Volumes*, vol. 45, no. 8, pp. 299–304, 2012.

[23] J. J. Craig, *Introduction to robotics: Mechanics and control*, 3rd ed. Upper Saddle River, NJ: Pearson, 2004.

[24] "ViperX-300 5DOF — interbotix x-series arms documentation," https://www.trossenrobotics.com/docs/interbotix_xsarms/specifications/vx300.html, accessed: 2022-11-10.

[25] "Wave loads," DNV AS, Class guideline - DNV-CG-0130, Oct. 2021.

[26] "Vessel theory: RAO quality checks," https://www.orcina.com/webhelp/OrcaFlex/Content/html/Vesseltheory,RAOqualitychecks.htm, accessed: 2022-11-19.

[27] V. E. Kremer, "Quaternions and slerp," *University of Saarbrücken, Department for Computer Science, Seminar Character Animation*, 2008. [Online]. Available: https://www.academia.edu/download/47081754/lerp_slerp_nlerp.pdf