# Autonomous Non-destructive Remote Robotic Inspection of Offshore Assets

Vladimir Ivan, Arnau Garriga-Casanovas, Pouyan Khalili,
Wolfgang Merkt, Frederic B. Cegla, Sethu Vijayakumar

**Abstract**

Offshore assets suffer from material degradation over their lifetimes. Regular inspections are necessary to prevent failures and to reduce the cost of maintenance. These often require downtime of the asset and can involve risk to human workers who have to be sent to the offshore location. In this work, we present a non-destructive (NDE) system in conjunction with a robotic platform, which can perform inspections of the thickness of a component, for example from the outside of a tank or a pressure vessel. The NDE system consists of a digital acquisition system and an electromagnetic acoustic transducer (EMAT). The EMAT generates an acoustic wave, which reflects from the internal features of the component. The wave is received by the same device. The received signal is then processed by the acquisition system to determine the thickness of the component. The NDE system is integrated with a robotic platform that can autonomously or semi-autonomously perform scans of the asset. The robot platform presented in this work uses sensor fusion, machine vision and state of the art motion planning techniques to build a map of the material quality in 3D. This is achieved by exploiting the precise movement of the robot end-effector along the surface of the asset and then integrating the position of the NDE sensor. The collected data is then presented to the remote operator in a user-friendly way, which allows them to evaluate the state of the asset. We validate this system using material samples with known defects. We performed experiments in a controlled environment, and we demonstrate the system in a case study at a testing facility operated by our industrial partners.

Fig. 1. Robotic non-destructive tank and pipe inspection using a mobile robot. The operator can inspect parts of the asset remotely and without decommissioning the asset for the duration of the inspection.

## I. Introduction

Offshore assets such as oil rigs are subject to different types of failures, some of which can be avoided through preventive maintenance. This requires human workers to travel to the remote location, incurring costs and potentially putting human lives in danger. Weather conditions may make it impossible to enter or leave the asset, further complicating the operations. It is also common to leave the asset unmanned for extended periods of time in which case a worker has to be dispatched to the asset every time an inspection is required.

With the onset of robotics, a new approach is emerging. It relies on the use of robotic systems and Artificial Intelligence to revolutionise Asset Integrity Management for the offshore energy sector. The aim is to equip existing and future assets with sensors and inspection robots that can operate and interact safely in autonomous or semi-autonomous modes in complex and cluttered environments. Human-machine systems will be able to co-operate with remotely located human workers through an intelligent interface that manages the cognitive load of users in these complex, high-risk situations. Full autonomy and remote operation modes will then enable inspection when no humans are on the asset. The robotics solutions will enable navigation around, accurate mapping of, and interaction with offshore assets. Robots and sensors will be integrated into a broad asset integrity information and planning platform that supports self-certification of the assets. This will enable the maintenance service providers with tools to evaluate and certify the state of the asset to their clients.

Vladimir Ivan, Wolfgang Merkt and Sethu Vijayakumar are with the Institute for Perception, Action, and Behaviour, School of Informatics, The University of Edinburgh (Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom).

Arnau Garriga-Casanovas, Pouyan Khalili, and Frederic B. Cegla are with the Department of Mechanical Engineering Imperial College London (City and Guilds Building, South Kensington Campus, London SW7 2AZ, United Kingdom).

Email: v.ivan@ed.ac.uk.

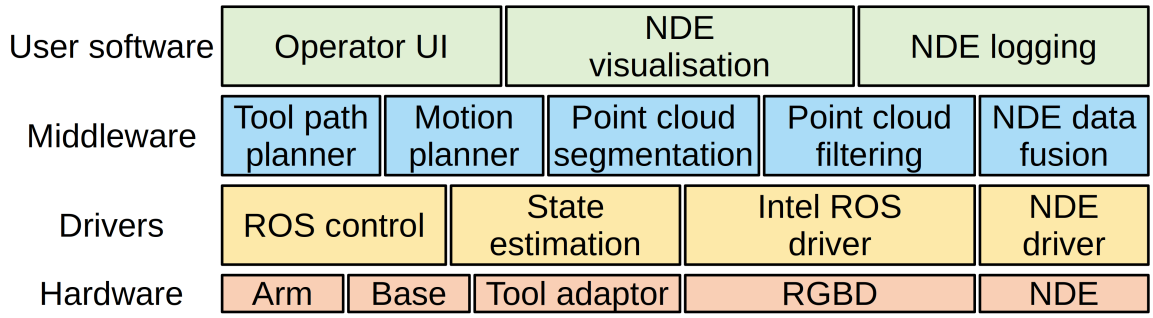


Fig. 2. Hardware and software components of the autonomous non-destructive inspection robot.

In this paper, we focus specifically on inspection tasks that involve detecting material failures in critical parts of the asset, such as pressure tanks and pipes. Our work enables us to perform periodic inspections remotely and with high levels of autonomy. This means that workers do not have to travel to the offshore location for routine inspections. Instead, the inspection crew collects data about the asset from a safe location on the shore. The data gets presented to the system operator in form of a map that is easy to understand and interpret. Additionally, the sensing technique we propose is non-destructive. This allows for shorter downtime due to the ability to inspect assets such as a pressure vessel from the outside, without the need to enter the vessel.

### *A. Related work*

Since visual inspection is not possible from the outside of a vessel, we focus on tracing the shape of the vessel with a non-destructive (NDE) sensor. A vision system can then be used to model the shape of the scanned surface by solving the problem of parametric model estimation from visual data such as point clouds. Point clouds can be obtained from stereo cameras or integrated RGBD[1] sensors. For point cloud data, RANSAC [Fischler and Bolles, 1987] and the many variants of this algorithm [Chum and Matas, 2005], [Kang and Li, 2015], [Torr and Zisserman, 2000] are the most widely used methods for shape detection. In more recent work, neural networks have also been applied to solve the shape detection problem in both supervised [Zou et al., 2017] and unsupervised [Sharma et al., 2017], [Tulsiani et al., 2016] settings. However, these methods have limited accuracy, the number of supported shape types is limited and extending the techniques for new types requires further re-training of the network. In either case, the result is a parametric shape that can be used to plan a path for a non-destructive sensor for inspecting the material that it is in contact with.

There are a wide variety of NDE techniques that can be employed to explore the internal components of a structure beyond the visual appearance that can be perceived with cameras [Hellier, 2012]. Most commonly, these techniques can be split into further sub-categories, based on the physical principles that are employed to reveal sub-surface information. The most commonly employed testing principles are radiography, ultrasonics and electromagnetic. In this work we focus on ultrasonics [Cheeke, 2016] and in particular the use of electro-magnetic acoustic transducers (EMATs) [Hirao and Ogi, 2003]. This is because EMATs do not require couplant which the robot would have to carry in an on-board reservoir. Furthermore, the wavefield and hence the type of inspection that is performed by an EMAT can be simply changed using different coils within the transducer head [Luo and Rose, 2003], [Hirao and Ogi, 1999], [Clough et al., 2017]. This ensures that we can carry out different inspections with minimal changes to the overall robotic system. The key drawback of EMATs is that they have a relatively low sensitivity and therefore there is a constant battle to ensure sufficient signal to noise ratio (SNR) to perform good measurements. The system that was used in this work was carefully designed, taking into account probe and driving electronics so that good results can be obtained.

Robotics researchers have been for some time exploiting the open source robotics ecosystem (ROS [Quigley et al., 2009]) and the availability of mature, standardised hardware platforms and software systems (e.g., ROS-Control [Chitta et al., 2017] for hardware abstraction and MoveIt! [Chitta, 2016] for motion planning). The industrial use of these software tools is also on the rise as the demand for industrial applications increases.

However, most of these tools treat the locomotion/navigation and manipulation problems separately and they only get joined and coordinated by a high-level state machine [Carius et al., 2018], sequence planner, or shared autonomy control interface [Fallon et al., 2015], [Merkt et al., 2017]. Such approaches limit the use of the robot to handle only static targets and obstacles and they disregard the inherent redundancy of high degree of freedom (DoF) systems. Yet, to achieve time- and energy-optimal solutions, locomotion and manipulation need to be considered jointly for the whole body of the robot.

[1]RGB stands for the colour data component (red, green, blue) and D in the acronym stands for the depth.

In response to this, [Hootsmans et al., 1992], [Yamamoto and Yun, 1992], [Kim et al., 2019] focus on whole-body control to coordinate and compensate end-effector motion. Such controllers then rely on input from motion planners such as [Dong Hun Shin et al., 2003], [Dang et al., 2011], [Yang et al., 2018], [Merkt et al., 2019a]. The above work focuses on loco-manipulation planning over longer horizons. The common theme is to optimise the robot motion subject to constraints such as tool position, collision avoidance and robot balance.

Constrained optimization is a common tool for planning and controlling motion of humanoid robots, as proposed by [Fallon et al., 2015] for trajectory optimization and for quadruped robots as proposed by [Xin et al., 2018] for task-space inverse dynamics-based control. In both cases, the task is formally described by constraints on controls, end-effector positions, and other properties such as static balance derived from inverted pendulum dynamics while control effort or other cost terms are minimised. A variety of efficient quadratic programming [Ferreau et al., 2014], [Andersen et al., 2013] and nonlinear programming (NLP) solvers [Gill et al., 2005], [Meza et al., 2007], [Wächter and Biegler, 2006] have been developed to solve these problems. These formulations are generic so that they can handle complex problem settings required for optimizing the motion of a highly dynamical legged system. We believe that these powerful tools should be adapted from legged systems that are still at the stage of proving their utility for industrial applications to commonly used mobile manipulation platforms and robot arms used across different industries, as we discussed in [Merkt et al., 2019b] and previously implemented in [Ivan et al., 2019]. This is the design approach we applied to the asset inspection task in this paper.

### *B. Contribution*

We propose a design of a robotic system that can inspect assets remotely. We combine autonomy provided by a mobile robot with a robot arm and a motion planning system that an operator can use to specify which parts of the asset to inspect. This system builds on fusing together the robot location, kinematics of the arm and point clouds captured by a RGBD camera to create a 3D model of the inspected component. We then use a novel type of sensor based on the EMAT technology to detect material thickness to identify material failure points. The novelty of the sensor is in the design that requires only several Watts of power, compared to the devices on the market with power consumption several orders of magnitude higher (kW). This type of sensing allows us to present the operator with a map of the asset that they can analyse. The key advantage of such inspection is that assets such as pressure vessels and pipes can be inspected without scheduling downtime and entirely remotely.

The contributions of this paper are: 1) the application of an EMAT type sensing system that is modular and straightforward to integrate with robotic systems, 2) the evaluation of the sensor performance used with a robot, 3) the integration of a novel non-destructive thickness mapping system, 4) the use of sensor fusion to enable inspection of components without prior knowledge of their shape (e.g. a CAD model), and 5) the evaluation of the system for an offshore asset maintenance case study. The novelty of this work for remote asset inspection is in the system design and integration.

## II. System design

We have integrated several hardware and software components to build an asset inspection robot. See Fig. 2 for an overview of the system components.

### *A. Hardware*

The robot is composed of an omni-directional mobile base[2] and a 6 degree of freedom (DoF) robot arm[3]. The mobile base houses the drive mechanism and electronics, control and sensing PCs[4], an inertia measurement unit (IMU[5]), and a battery. The wheel motors are velocity controlled at 1KHz control rate. The robot arm is position controlled at 1KHz. These specifications allow us to track a base trajectory with approximately 3cm position drift over 1m travel distance without using any exteroceptive sensing for feedback and approximately 1cm end-effector position error while the base is moving. The end-effector accuracy is sub-millimetre when the base is stationary. See [Merkt et al., 2019b] for more details.

We have attached an Intel RealSense D435 (RGBD) camera to the end-effector of the robot. We also equipped the end-effector with a compliant mechanism. This mechanism is a spring loaded linear slider with a position encoder to detect the sensor displacement on contact with the scanned component. The sensor we used is a non-destructive material thickness sensor we designed. We will discuss the NDE sensor in the next section. This sensor requires a contact with the scanned sample, which is why we designed a compliant attachment for it. See Fig. 4 for overview of the hardware components.

[2]Ada500 omni-directional mobile base.
[3]Han's Robot Elfin5 collaborative robot arm.
[4]Intel NUC with i7-7567U (3.5GHz and 4MB cache) processor and 8GB RAM.
[5]Lord Microstrain 3DM-GX5-25

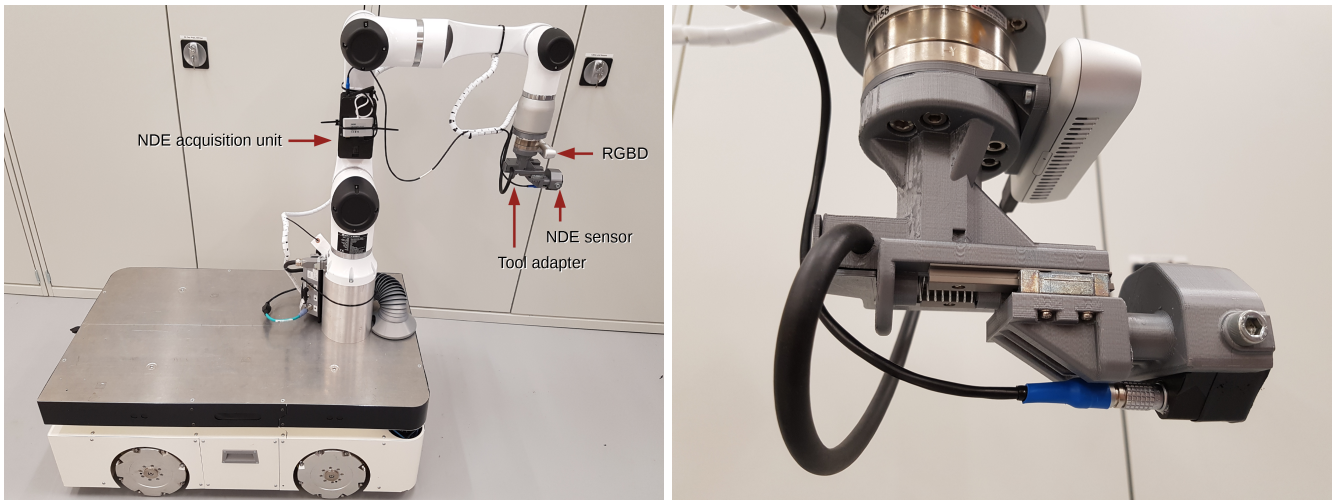


Fig. 3. The omni-directional robot base (left) houses the drive, battery, on-board PCs, and the robot arm is mounted at its front. The NDE acquisition system is mounted on the robot arm. The RGBD camera is attached to the fixed part of the tool adapter while the NDE sensor is attached to the compliant end. Right: the close up of the compliant tool adapter.

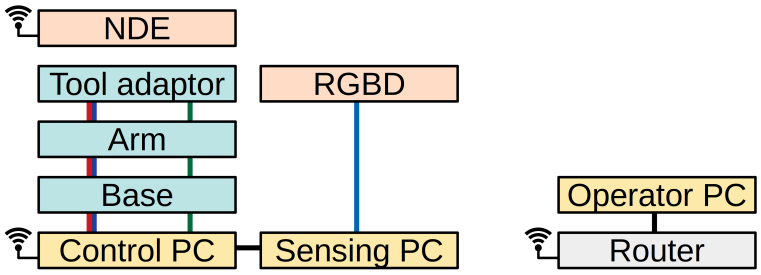


Fig. 4. The robot arm, base and the tool adaptor are connected to the control PC (red/blue - power, and green - ethercat). The RGBD camera is connected to a dedicated sensing PC on board. This PC is connected to the control PC via ethernet (black). The NDE sensor is electrically separated from the rest of the robot (it runs on its own battery). The NDE sensor and the control PC are connected to the operator station via a WiFi router.

### *B. Low-level software and drivers*

The robot base and arm use the ROS control[6] framework to expose the robot state (position and velocity of each joint and of the wheels) and to expose the command interfaces (velocity commands for the base and position commands for the arms). The ROS control framework then provides controllers for tracking trajectories and for streaming position references to the arm and the base. The control is interfaced with a state estimation module that computes the robot position in the world and the locations of sensors. This system architecture is similar to the one proposed in [Merkt et al., 2019b]. We will discuss the state estimation module in Section IV.

The RGBD camera drivers are provided by the manufacturer. The NDE sensor drivers are proprietary and developed by our team.

All the low-level software components are either directly written or wrapped in ROS. This provides an unified interface for the middleware that uses this low-level software layer.

### *C. Middleware*

The middleware layer consists of a point cloud filtering tool that produces a single consistent point cloud at a desired resolution and a point cloud segmentation tool that fits planar or cylindrical surfaces to the point could data. This layer also consists of a tool path planner and a motion planner for generating the robot motion. We also developed a calibration tool to calculate the joint and camera offsets to improve the accuracy of the pipeline. The last part of this layer is the NDE sensor data fusion module that creates the thickness maps of the scanned components. We will discuss the NDE sensor data fusion, the point cloud processing, and the planning modules respectively in Sections III, IV, and V.

[6] `http://wiki.ros.org/ros_control`

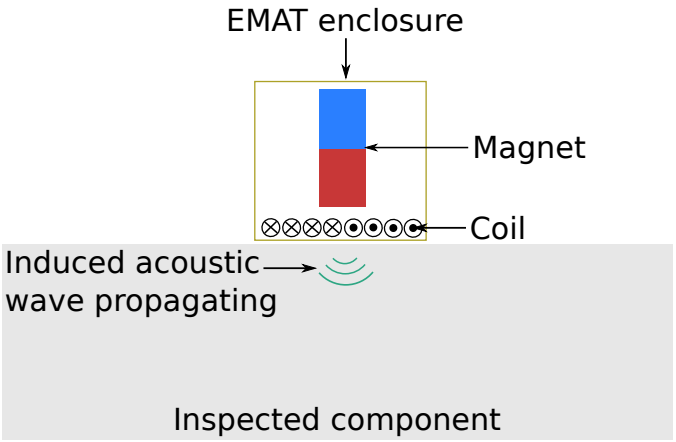


Fig. 5. Schematic diagram of a typical EMAT.

### D. User interface

The user of the system interfaces with the robot remotely. We use the RViz[7] visualiser to display the robot, the point could, and the segmented surfaces. We also used an interactive marker[8] to design a simple interactive widget to select an area to scan and to trigger the scan itself.

We then use a separate visualisation window to display the 3D thickness maps produced from the NDE sensor data. We also use a separate data logging tool, rosbag[9].

We will now describe the individual sub-systems in detail.

## III. NDE SENSING

The sensor used in this work belongs to the category of sensors known as electromagnetic acoustic transducers (EMATs), which are transducers that rely on electromagnetic induction to excite and receive ultrasonic waves on conductive components [Hirao and Ogi, 2003]. In general, EMATs consist of two key components, which are a magnet and a coil. The magnet is usually permanent, and the coil is excited with an electrical current. A typical diagram of a standard EMAT is shown in Fig. 5.

The coil in the EMAT induces eddy currents in the inspected component when the coil is in its proximity. To achieve this, the coil is excited with a pulse of electrical current. The eddy currents induced in the component interact with the magnetic field from the magnet in the EMAT, thereby generating a Lorentz force. This creates a strain in the material of the inspected component. This strain then starts an acoustic wave, which propagates and reflects on any internal features of the component. The reflections are then received by the same EMAT through an analogous mechanism. We measure these signals as a voltage between the ends of the coil. We then process the received signals to extract the information of interest, which can be for example the thickness of the component.

EMATs can generally be used for two main inspections purposes: either thickness measurement or crack detection. In the first case, the thickness of the component is determined based on the time of flight in the arrival of the back wall echoes. In the second case, the presence of a crack can be inferred by processing the amplitude of the waves received in two different coils that induces shear waves in perpendicular directions [Ribichini et al., 2012].

EMATs can conduct inspections without direct contact with the component of interest, since they employ electromagnetic induction to generate the acoustic waves. The typical separation between the coil and the conductive material for our sensor (as well other EMAT designs) ranges between 0mm and 2mm. This enables inspections of components with coatings such as paint, and inspections of components with surface roughness (which can be due to corrosion) without the need for preliminary surface preparation. In addition, EMATs can be used for the inspection of hot surfaces since they do not require a coupling nor direct contact with the surface. Additionally, this technology can be easily used to scan components because it does not require a coupling medium nor surface preparation. As a result, EMATs are appropriate to be used for robotic inspections.

### A. Design of sensor and acquisition system

The specific sensor used in this work consists of a permanent magnet arranged to produce a magnetic field in an equivalent manner as in Fig. 5, and a set of two coils connected together, which lay in a plane parallel to the surface of the inspected

[7] http://wiki.ros.org/rviz
[8] http://wiki.ros.org/rviz/Tutorials/Interactive%20Markers%3A%20Getting%20Started
[9] http://wiki.ros.org/rosbag

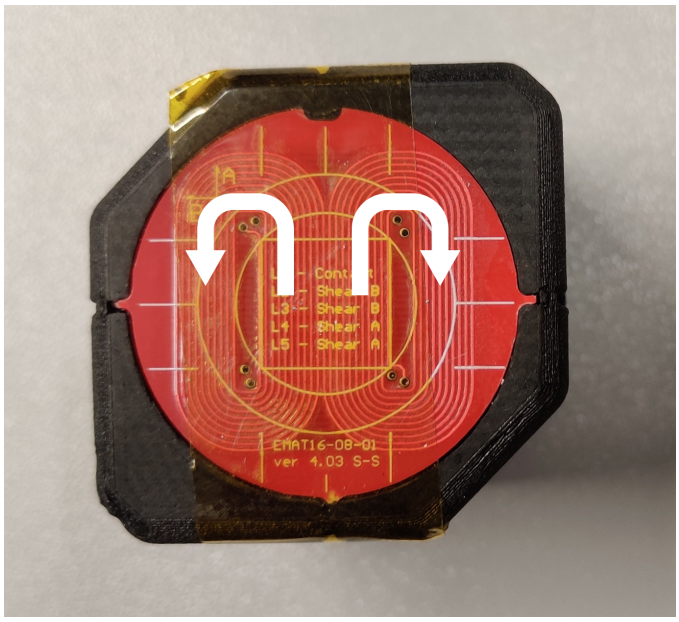


Fig. 6. Bottom part of an EMAT probe with the coil.

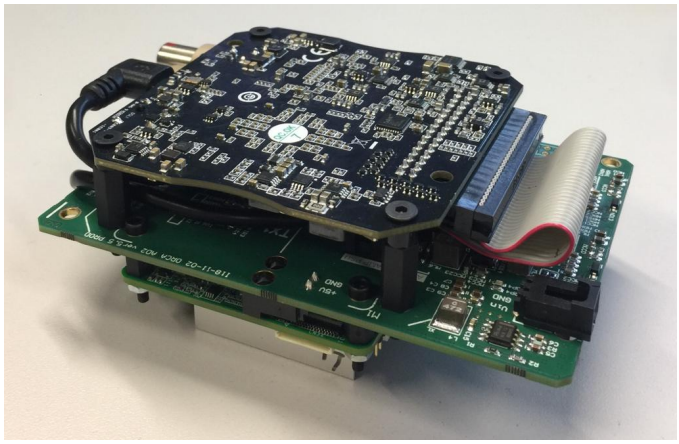

Fig. 7. Acquisition system for EMATs comprising the main elements: a processor, an arbitrary wave generator and oscilloscope, and a PCB with amplifiers and power regulation.

component. Fig. 6 shows the bottom of the sensor with the coils. The coils are wound in opposite directions, as indicated in Fig. 6, such that a current passing through the coils induces opposite magnetic fields in both of them. This leads to an induced current in the component that is directional, following the direction between the centres of the two coils. The result is that the strain in the material, and thus the shear wave generated by the EMAT, is polarised in a specific direction.

This sensor can be used for the inspection of the thickness of components made of typical materials found in industry, such as steel or aluminium. The coil in the sensor has a diameter of approximately 3cm. The permanent magnet on the sensor produces a magnetic force of approximately 40N when placed on mild steel. The sensor can be scanned over the surface of a component using a robot without the need for any rollers to obtain a thickness map of the component.

This EMAT sensor requires an acquisition system to excite it with a current pulse, and to read the resulting signals from it. We have developed a new acquisition system which can be used with practically any EMAT sensor.

The acquisition system is shown in Fig. 7, and consists of a processor, a PCB with amplifiers and power regulation, and an arbitrary wave generator and oscilloscope. The arbitrary wave generator can produce arbitrary shaped waveforms with frequency content between 100KHz and 5MHz, and it is used to excite the EMAT sensor. The signals received from the sensor are then amplified by analogue amplifiers in the PCB. We custom designed the amplifier to provide signal amplification of approximately 70dB. The processor controls the wave generator and oscilloscope, it performs the required signal processing, and it transmits the data over WiFi. The system is wireless, and it is powered by an on-board battery. The power supply from the battery is regulated by the power regulation on the PCB to supply the required input to the different elements.

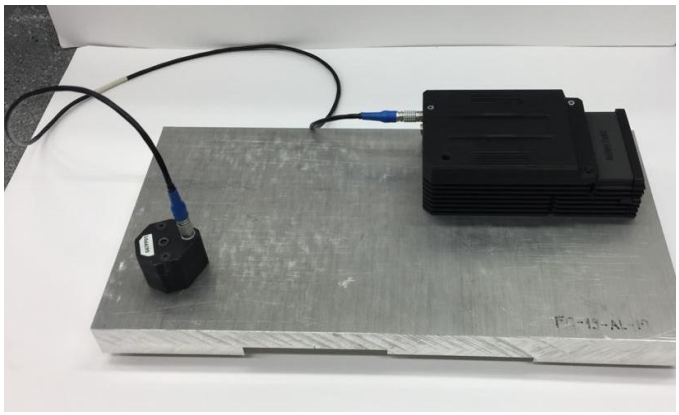

Fig. 8. Acquisition system assembled inside an enclosure, with battery and connected to an EMAT probe.

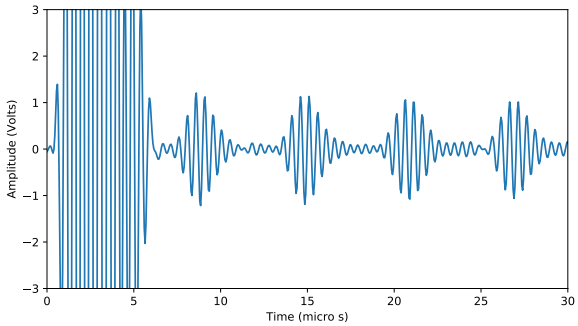


Fig. 9. Example of resulting time-trace obtained in a test with the EMAT probe on a 10mm thick mild steel plate.

We designed the acquisition system to be small, modular, lightweight, and versatile, so that it can be easily used with any robot. It weighs 140g excluding the battery, or 310g including a 2.6Ah battery and an enclosure box. It can be assembled into an enclosure of size 145x85x43mm, as shown in Fig. 8. It is standalone, and it can perform continuous inspection for approximately 1.5h using the 2.6Ah battery. It also uses low power electronics of a maximum of 15W, and it can be used with practically any EMAT probe.

### *B. Inspection parameters*

In this paper we use the acquisition system and the EMAT sensor to measure thickness of components. For this, we operate the acquisition system at a frequency of 2MHz. We selected signal sampling frequency of 20MHz.

In order to obtain each thickness measurement, we average over 10 acquisitions to obtain a noise free time-trace of the ultrasonic wave. The number of averages is a parameter that can be increased in more noisy conditions. We then process the time-trace to determine the thickness of the component based on the time difference between the peaks of the second and third echo from the back wall reflection. An example of resulting time-trace obtained in a 10mm thick component made of mild steel is shown in Fig. 9. To determine the thickness of the material from the time-trace, we need to know the propagation velocity of the wave. In this work, we determine the velocity of the shear waves in the material by calibrating with respect to a known reference sample thickness.

The system runs the Robotics Operating System (ROS) [Quigley et al., 2009]. We use ROS to implement the NDE driver node which transmits the data to either a robot or another receiving platform such as a laptop. The NDE driver then feeds the thickness data into the middleware that performs sensor fusion and builds the maps of the component.

## IV. State estimation and surface modelling

All the middleware modules rely on accurate state estimation and availability of the location of the camera, base and NDE sensor frames at the time of data acquisition. To achieve this, we integrated a multi-modal state estimator implementing the Unscented Kalman Filter algorithm (UKF), [Wan and Van Der Merwe, 2000]. The state estimator takes several inputs: position and orientation from the robot odometry, and angular acceleration from the IMU. Our previous experiments that we performed in [Merkt et al., 2019b] have shown that the UKF algorithm performs marginally better than the Extended Kalman Filter (EKF) presented in [Moore and Stouch, 2014] which is also commonly used for this task. We have also

experimented with using the linear acceleration from the IMU but this input decreased the accuracy of the state estimator. The linear acceleration (and velocity) data is known to exhibit drift in most commercial grade IMUs. This is because a highly accurate gyroscope is required to calculate the direction of gravity vector that has to be subtracted from the measured linear acceleration. In our experience, only fibre-optic gyroscopes have the required accuracy. Nonetheless, the odometry and IMU provided an accurate enough estimate of the robot's location for point-to-point navigation tasks. The output of this module is a transformation of the robot base with respect to an odometry frame[10]. We used the implementation of the UKF algorithm from the *robot_localization* ROS package[11].

We then combined the robot base transformation with a forward kinematics module. This module computes the relative frames of the sensors from the kinematic model of the robot and the joint angles of the actuators. The robot model is specified using the *Unified Robot Description Format* (URDF[12]). The URDF model is stored in an XML file and it provides the locations of the robot links, the actuators, and the sensors. This file is then loaded into a kinematics solver which computes the sensor frame locations using joint encoder readouts. It keeps the time stamp of the joint angle reading and builds a buffer of past sensor frame locations. This enables us to synchronise sensor locations with sensor readouts even if they are not taken at the same time. We use the implementation from the ROS TF2[13] package. However, to get an accurate sensor position, the robot model has to be calibrated.

### *A. Kinematic calibration*

To collect data samples and build an accurate and consistent thickness map, we require the exact position of the sensor at the time when the data was captured. Since we have chosen to perform scans only when the robot base is static, we only need to worry about the accuracy of the tool frame with respect to the robot base frame. Still, there are several possible sources of kinematic error for the tool frame: accuracy of the assembly of the robot arm (position on the base, offset of the joint encoders), imprecise mounting of the RGBD camera and NDE sensor, and the RGBD camera optics.

We have used a RGBD camera that is pre-calibrated by the manufacturer. This ensures that the intrinsic parameters (lens distortion and the camera projection matrices) are used when computing the point cloud from the visual data. The camera driver takes care of this task and it outputs a point cloud with position accuracy between 2.5mm and 5mm at the distances we perform the scans. See Fig. 11 for an example of a 3D point cloud obtained during the experiments. To ensure that the point cloud is consistent when we view the same component from different angles, we need to calibrate the camera extrinsic parameters: the position of the camera relative to the robot base frame. The camera is mounted on the tool adapter at the end of the arm. To eliminate any errors introduced during the assembly of the tool adapter, we choose to calibrate for the camera frame offset relative to the robot arm tool flange.

Additionally, the robot arm we use has a known issue with the joint angle zero position shifting slightly due to vibrations during transport. For this reason, we chose to calibrate the joint angle offsets as well. Our calibration procedure starts by collecting the calibration data as described in Algorithm 1.

**Algorithm 1:** Kinematic calibration data collection

**Result:** $Q$, $T$

$Q \longleftarrow \emptyset$

$T \longleftarrow \emptyset$

**foreach** *Arm pose $q$ from calibration poses* **do**

  $MoveTo(q)$

  $CapturePointCloud()$

  $RemoveOutliers()$

  $Q \longleftarrow Q + \{q\}$

  $T \longleftarrow T + \{FitPlaneToPointCloud()\}$

**end**

The data is collected by moving the robot arm to various poses where the top plate of the robot base is visible. We have manually chosen 20 robot poses that view the top plate from different angles and that use different joint configurations. We aimed to empirically maximise the joint angle variance between the poses. As we move the robot arm through these poses, we collect a data set of joint angles $Q$, and a data set of plane coordinates $T$ which describe the slope of the top plate of the robot relative to the camera. To obtain the plane coordinates, we use the Random Sample Consensus algorithm (RANSAC) to fit a plane model, [Fischler and Bolles, 1987]. To remove outliers and reduce the size of the point cloud, we remove points that lie outside of a predefined bounding box. In our case, we transform the point could to the base frame,

[10]Odometry frame is an arbitrarily chosen frame fixed with respect to the world, e.g. the location of the robot when it was turned on. It should be distinguished from a map frame, which is the globally consistent origin of the map, e.g. the location of the docking station.

[11]https://github.com/cra-ros-pkg/robot_localization

[12]http://wiki.ros.org/urdf

[13]http://wiki.ros.org/tf2

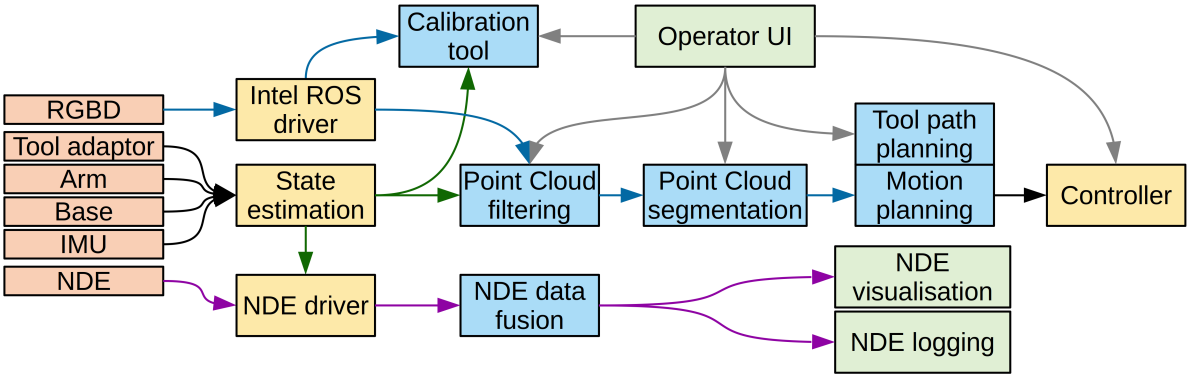


Fig. 10. This diagram shows how different modules of the system interact with each other. The joint encoder values and IMU data (black arrows) is passed to the state estimation. The controller also receives the joint angle trajectories as the command input. Point cloud data (blue arrows) are passed through the sensing pipeline all the way to motion planning. NDE data (purple arrows) are passed through sensor fusion into visualisation and logging. The user interface interacts with various modules to allow the user to trigger actions such as point cloud segmentation, planning, and execution of the scan.

define the bounding box around the top plate of the robot base, remove the outliers, fit the plane and then transform the plane coordinates back to the camera frame. We use the uncalibrated camera frame for this process as an initial guess. Even though the camera frame may still contain the calibration errors at this stage, it is sufficient enough for outlier removal. We use the RANSAC and box filtering implementation from the Point Cloud library (PCL[14]). Furthermore, we use the PCL library for implementing all the point clout filtering and segmentation algorithms described in this section.

We then use the joint poses $Q$ and plane parameters $T$ to minimise the projection error of viewing the robot top plate plane from different angles. The assumption is that a set of optimal joint angle offsets and a camera frame offset will make the plane of the top plate look consistently flat from any viewing angle. We start by defining the variables we optimise over. These are the joint angle offsets $q_0$ and the camera frame offset $T_{camera}$ parametrised as a vector of positions $xyz$ and roll, pitch, yaw angular orientation offsets. The resulting vector of variables is then $x = [q_0, x, y, z, r_{roll}, r_{pitch}, r_{yaw}]$, where $q_0 \in \mathbb{R}^6$ because our robot arm has 6 degrees of freedom. We then formulate the optimisation as:

$$\underset{x}{\operatorname{argmin}} \sum_i \rho_1 f_{normal}(T_i)^2 + \rho_2 f_{plane}(T_i)^2 \tag{1}$$

$$where \quad T_i = \Phi(q_i + q_0) * Transform(x, y, z, r_{roll}, r_{pitch}, r_{yaw}) \tag{2}$$

where $Transform()$ constructs a transformation matrix from the position and Euler angles, $\Phi()$ computes the forward kinematics using the robot model and the joint angles $q_i + q_0$ and outputs the camera frame with respect to the top plate frame, $f_{normal}(T_i)$ computes the plane normal axis alignment error, and $f_{plane}(T_i)$ computes the plane origin distance. $\rho_1$ and $\rho_2$ are scalar weights we use to adjust the relative scaling and steepness of the cost function[15]. We set the scaling factors to $10^4$ and $10^6$ respectively. $T_i$ is the frame representing the centre of the plane segmented from the point cloud with the $z$ axis of the coordinate system aligned with the plane normal. We then compute the axis alignment error by comparing the $z$ axes of the $T_i$ frame with the unit vector pointing in the positive z direction: $f_{normal}(T_i) = 1 - T_{i,z} \cdot [0, 0, 1]$. We also penalise for distance of the centre of the plane from the top plate of the robot base using $f_{plane}(T_i) = T_{i,z}$. We implement the minimization using the Python SciPy least squares minimisation algorithm. The forward kinematics are computed using the EXOTica library, [Ivan et al., 2019]. The data collection takes around 1 minute to complete and the algorithm takes several seconds to optimise the calibration parameters. We use the calibration offsets to update the robot model. This calibration is only required before the inspection task begins and only after the robot has been transported or mechanically modified. Hence, the calibration time is negligible. In our experiments we achieve calibration accuracy of 0.1mm, measured as the residual error using the $f_{plane}(T_i)$ function (the distance of the segmented plane centre from the top plate of the robot). We deem this accuracy sufficient since it is below the depth resolution of the RGBD camera as well as the resolution of the NDE sensor. The calibrated camera and NDE sensor frames are then used in the sensing pipeline. See Fig. 10 for the overview of how the modules interact.

[14] https://github.com/PointCloudLibrary/pcl

[15] Even though absolute scaling of the cost function should not affect the optimisation, in practice, we find that we have to scale the cost terms relative to the solver precision threshold for best performance. All terms of the cost function also have to be scaled relative to each other to ensure desired accuracy trade-off between the orientation and position error.

### B. Surface reconstruction from point clouds

We have decided to split the point cloud processing pipeline into two separate modules: 1) point cloud filtering and aggregation, and 2) surface segmentation. The first module ensures that we have a consistent point cloud for use in the second module. To achieve this, we perform the following steps: 1) move to a predefined scanning pose, 2) outlier removal (we use the same bounding box method that we used for the calibration routine), 3) transform the points into the world frame (using the frame transformation from state estimation), 4) aggregate the point cloud, and 5) reduce the density using voxel grid filtering. All of the operations are implemented using the PCL library[14]. We move the camera so that we can view the sample in front of the robot with enough coverage. We have manually defined a sequence of arm motions to achieve this. By following the steps above, we capture a point cloud at each pose and add it to the aggregated point cloud. We can simply copy the newly captured points into the aggregated cloud since all points are already transformed into the world frame.

The point cloud filtering module has two parameters: the size of the bounding box, and the desired resolution/density of the point cloud. We have chosen a bounding box that limits the viewing distance of the camera to 1.5m (the range at which the resolution of the camera is sufficient for our task) and a point could resolution of 1mm. The aggregated point cloud is then passed onto the surface segmentation module.

The aim of the surface segmentation module is to fit a parametric surface model to the point cloud so that we can use this model for planning the tool path. We are currently fitting either a plane or a cylinder surface. The user gets to choose which surface parametrisation to use. Algorithm 2 shows the segmentation procedure.

**Algorithm 2:** Surface segmentation

```
1  Remove points outside of user defined bounding box
   if Plane segmentation then
2      Segment plane using RANSAC
3      Correct plane normal to point towards the robot
4      Project centre of the point cloud onto the plane
5      Calculate the planar segment size
6      Calculate the transformation matrix
   end
   if Cylinder segmentation then
7      Estimate point normals
8      Segment cylinder using RANSAC
9      Correct cylinder axis to point perpendicular to the viewing direction
10     Project centre of the point cloud onto the cylinder axis
11     Calculate the cylinder segment size
12     Calculate the transformation matrix
   end
```

The segmentation is triggered by the user selecting a bounding box around the sample in the user interface. This bounding box is used on line 1. We use the RANSAC algorithm to segment the plane (line 2) as well as the cylinder (line 8). The cylinder segmentation additionally requires the local surface normal to be estimated for each point. These are calculated on line 7). We define a convention for the surface orientation. For both types of surfaces the surface normal points towards the robot (line 3). For the cylinder we approximately align the $y$ axis with the viewing direction (line 9) while the $z$ axis along the height of the cylinder is estimated using RANSAC. We then compute the segment centre and project in onto the plane (line 4) and the cylinder $z$ axis (line 10). This ensures that we can display consistently aligned segmented surfaces if we need to re-run the segmentation algorithm.

We also compute the size of the segment by fitting a bounding box around the inliers we obtain from the RANSAC algorithm. The output of the algorithm is the size of the segment and the transformation matrix we compute from the orthogonal axes of the segmented surface (e.g. the plane normal and the cylinder central axis). See Fig. 11 for the example output of the point cloud processing pipeline.

The segmentation module has several parameters that are required for the RANSAC algorithm: distance threshold for inlier detection, number of iterations, minimum and maximum cylinder radius, and point normal normal approximation search radius (used on line 7). We have empirically set these parameters based on the requirements of our task. The key deciding factors were the size of the sample, required accuracy of the surface fit, and available computation time. We then use the parametrised surface models in the motion planning module to constrain the tool path to the surface of the sample.

## V. Motion planning and control

The purpose of the motion planning module is to take the parametrised surface model and compute a trajectory for the robot arm to sweep the surface of the sample with the NDE sensor. This procedure has three steps: 1) the user selects the area to scan, 2) we plan the tool path, 3) we plan the robot motion.

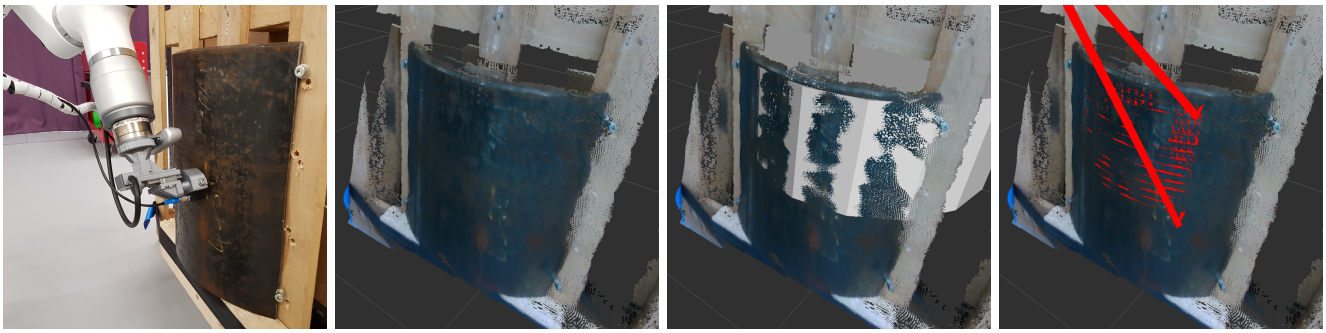

Fig. 11. Point cloud processing pipeline stages, from left: picture of the setup, captured point cloud, fitted cylinder model (white), tool path projected on the cylinder model (red). The tool path starts off the view at the current location of the tool. It then traces horizontal lines in a grid pattern before it returns back to the original position. Some of the path is rendered inside of the component due to noise of the RGBD camera. The compliant mechanism will deal with this error when the tool is in contact with the component.

The user interface for selecting the scan area is the same one we use to select the bounding volume for surface segmentation as described in Section IV-B. We reuse the same bounding box to define the scanned area. In an ideal scenario, the operator would request a point cloud scan, select the desired region to inspect with the NDE sensor and then trigger the scanning process. We call this shared autonomy and we adjust the level of human input based on the task and user preferences. At the request of our operators, we keep the human in the loop to approve each sub-task. The execution and data collection are then fully automated.

The user interface ensures that the bounding box is always aligned with robot base. This makes it easy to plan a tool path in 2D and to then project it in onto the segmented 3D surface. We use a simple grid scanning pattern that covers the selected rectangular area. We sample a 2D position along the trajectory, project it into 3D at the robot origin and intersect the ray with the parametrised surface. The ray is cast from the robot origin in the forward direction, as defined by the location of the bounding volume. We calculate the intersection point $p$ and surface normal $n$ on the parametrised surface. We then define an orthogonal tool coordinate system $T$ around the intersection point as follows:

$$T = \begin{pmatrix} n_x & n_y & n_z & p \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$\begin{aligned} n_x &= n_y \times n_z \\ n_y &= n_z \times n_{up} \\ n_z &= -n \end{aligned} \tag{4}$$

$$\tag{5}$$

where $n_{up}$ is a unit vector pointing in the desired tool up direction. We use this direction vector to orient the tool (NDE sensor) relative to the grain of the sample material. It is sometimes required to reorient the tool and re-run the scan if the material grain is aligned with the sensor orientation, as this causes decreased accuracy of the NDE sensor. The tool transformation matrix $T$ is the tool position relative to the robot base. We compute a trajectory of tool positions. We also ensure that the every tool target can be projected onto the parametrised surface. If it cannot, we skip the section of the scan until a consistent tool trajectory segment that can be projected is found. If such segment can't be found, we report a planning failure through the user interface. We then pass a valid tool trajectory onto the motion planner. See Fig. 11 right for example of a planned tool path.

The motion planning module implements a kinematic trajectory optimisation technique based on non-linear optimisation. We treat the robot model as a non-linear function calculating offset of the tool from the desired tool trajectory computed in the previous module. We extend the tool path to start and end at the current tool position. We also sub-sample the path and linearly interpolate between the existing trajectory knot points to achieve a higher tracking accuracy and motion smoothness. We then formulate a non-linear optimisation problem as defined in [Merkt et al., 2019b]. We use the EXOTica library [Ivan et al., 2019] to implement the motion planner. The result is a trajectory of robot arm joint angles that we execute using the joint position controller. If the planner fails to find a solution, e.g. the tool position and orientation are not reachable, we report the failure to the user via the UI.

The user interface allows the operator to start a camera scan of the area in front of the robot to create a point cloud. The operator then moves a widget around to select an area to scan with the NDE sensor. The operator then triggers the surface fitting and motion planning and, finally, executes the motion. While the robot is moving along the planned trajectory, the NDE sensor gets pressed against the sample and aligned with the surface. Once the compliant tool adapter is compressed

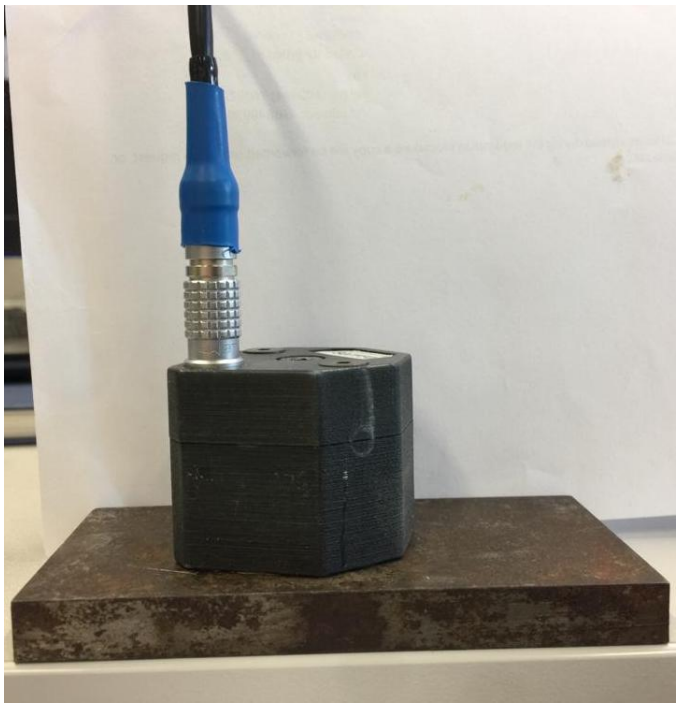

Fig. 12. Example of the set up for the initial evaluation of sensor performance, with the sensor manually placed on a steel component.

sufficiently to detect a stable contact, the NDE fusion module starts outputting tool position and material thickness data.

### A. NDE sensor fusion and the operator user interface

The NDE thickness data gets aggregated while the sensor is tracing a path on the surface of the sample. We use the position of the sensor obtained from the state estimator to record the sensed location, and the thickness from the NDE driver to build a 3D thickness map of the scanned component. Each point in the map contains the sensor location on the surface, far side location, and the thickness value. Once we have collected enough samples, we perform a Dalaunay triangulation on the sensor location points to create a mesh representation of the map. We then display this mesh to the operator in a separate visualisation window. We also colour-code the thickness of the material to make it easier to spot defects and thickness variations. For this, we choose a colour scaling that shows the expected thickness in blue and abnormal thickness in red, to alert the operator. See Fig. 13 right for the output of the sensor fusion visualisation.

## VI. Experiments

We have performed several experiments to validate this workflow. For our evaluations, we scan sample components with known shapes and thicknesses.

### A. Sensor performance evaluation

We began by characterising the performance of the NDE sensor and acquisition system separately from the robotic system. This ensures that the system is suitable to produce thickness data sufficiently accurate for building the thickness maps. The initial evaluation tests involved placing the sensor manually on a set of randomly chosen locations on a plate made of mild steel, and performing measurements in each location. The plate had a thickness of 10mm, which is representative of the thickness of the components we plan to map in the subsequent experiments. An example of the sensor placed on a steel plate to obtain a measurement is shown in Fig. 12.

The acquisition system was used to obtain time-traces in each of the randomly selected locations, and to process the resulting time-trace to obtain a thickness value. A representative time-trace obtained in one of the locations is shown in Fig. 9. The received time-traces were manually inspected to ensure that the signal appeared as expected, with an initial tone-burst and a set of echoes from the back wall, as the one in Fig. 9. The automatic processing of the signal performed by the acquisition system provided a thickness value, which was also checked to ensure that it correctly matched the plate thickness. We measured the test plate thickness using a caliper as a reference. In all locations, the thickness value determined by the system was correct, and time-trace signals appeared as expected.


Fig. 13. 10mm thick mild steel sample used for system validation experiments. Left: back side with a 2mm deep slot, middle: front side with yellow coating applied, right: thickness scan produced using our system in millimetres ($10^{-3}m$).

We measure the signal to noise ratio (SNR) in one representative time-trace. We determined the SNR based on the ratio between the maximum amplitude in one of the echoes and the maximum amplitude in the noise between the echoes. The SNR value obtained on mild steel was 25dB.

We measured the repeatability and accuracy of the sensor and acquisition system by placing the sensor in a representative location on a 10mm thick plate made of mild steel and performing 50 independent measurements of thickness, each averaged over 10 signal samples. The results of these measurements showed that the system correctly measured the plate thickness in all 50 cases at 10mm $\pm$ 0.08mm. This is considered to be a suitable accuracy for thickness inspections where an accuracy of 100 micrometres is typically required.

### *B. Robotic inspection system evaluation*

Having characterised the performance of the NDE sensor, we created an experimental setup for validating the system as a whole. For this, we prepared a 600x600mm steel plate with 10mm thickness on the back side of which we milled out a 2mm deep slot (see Fig. 13). This creates an 8mm thick patch that we aim to detect by scanning the front side of the sample. We have also coated the front side of the sample to emulate protective coating that is usually applied to protect the assets from corrosion.

We have performed a scan of the flat sample as described in the previous section. The NDE sensor swept a path on the surface of the sample during which the compliant tool adaptor ensured a stable contact. We have set the angle of the tool to 45° from the up direction. Several previous experiments with the upright tool orientation have shown decrease in performance. This is because the sensor coils were aligned with the grain of the steel. It is impossible to know the grain orientation when scanning an unknown sample. We therefore provide an option to change the tool orientation and re-run the data collection. With the correct tool orientation, we achieve the same thickness accuracy as reported in previous experiment.

We have recorded the thickness values and sensor positions and plotted the thickness map (see Fig. 13 right). The boundaries between the two thickness levels show higher levels of noise. We believe that this is due to acoustic effects around the sharp edges we milled into the sample material.

The density of the thickness map depends on the sensor acquisition rate and the tool speed. We can always decrease the tool speed to arbitrarily increase the map density. However, this will come at the cost of increased duration of the scan. We have experimented with several acquisition rates and tool speeds and we have settled at 0.02m/s tool speed and 5Hz acquisition rate. This provided us with 0.004m (4mm) sample density.

We have also repeated the experiment outside of the lab environment. We have deployed the system at an offshore research and development facility. The aim of this exercise was to show the robustness of the system in a novel environment. We have repeated the flat sample scan in this new environment. Fig. 14 shows the setup (left) and the results of this experiment (right). Notice the lower density of the map. This is because we could not shield the sensor from external sources of electrical noise as well as in the lab. As a result, we increased the number of samples to average over to 30 and decreased the acquisition rate to 3Hz to achieve more consistent thickness readings. With a lower acquisition rate but same tool speed the density of the map decreased as well. Alternatively, the density of the thickness map can be increased by commanding the tool to move slower. This may indeed be preferable in many scenarios where a more detailed inspection is required.

## VII. Conclusion

The system we developed combines the accuracy and autonomy of a robotic platform with novel EMAT type sensing system for the inspection of assets for material faults. Our experiments validate that we can detect material thickness changes in steel plates. We have shown a comparison on samples we have manufactured ourselves to provide ground truth for the experiments. In the future, we intend to test the system on sample materials that underwent corrosion on actual offshore assets.

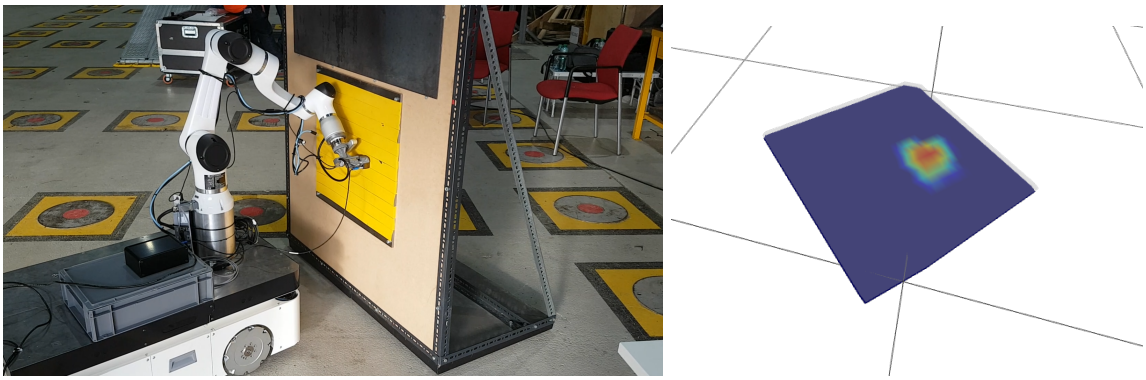

Fig. 14. Inspection task performed at a industrial site (left) and the obtained lower resolution scan of the 10mm thick mild steel sample (right). Since we display a 3D map of the component, we overlay the front side of the material as a semi transparent surface to visualise the thickness more clearly. This slightly distorts the colour map when enabled.

In this work, the NDE sensor is used to measure the thickness of the inspected component, using a single coil. This inspection configuration is somewhat sensitive to the orientation of the material grain, which we compensate for by planning a tool path at an angle that is the most desirable. However, it is possible to use the sensor with a dual coil configuration, which can be used to electronically select the orientation of the generated waves, and thus detect grain orientation and compensate for its negative effects. This change will also allow us to detect cracks and other more serious material faults using advanced signal processing tools. We provide a proof of concept. We will evaluate a range of thicknesses, materials, coating, and defects in our future work.

The integration of the NDE sensor with the robotic platform provided us with an accurate sensor position. This was crucial for building consistent thickness maps. The sub-millimetre accuracy ensures that the position along the surface of the sample is precise. The distance to the surface itself is handled by a compliant element that ensures we stay in contact with the surface during the scan. This is usually an issue only when scanning non-ferromagnetic materials such as aluminium. When working with steel samples, the permanent magnet inside sensor creates a strong pulling force that ensures a robust contact. However, this force also causes large friction forces when sliding the sensor along the surface. The robot arm can easily overcome these forces, but they cause vibrations and, in extreme cases, possibly cause damage to the sensor or the sample. We are working on protective coating for the sensor that will reduce its friction coefficient. This is especially important when dealing with samples with a rough surface, e.g. when the protective coat (paint) on the sample has been corroded away.

To some extent, the effect of the friction forces can be reduced by planning a motion that can exploit the knowledge of the surface shape. At the moment, we do this using the camera system by estimating the surface curvature. Our sensing pipeline implements plane and cylindrical surface fitting at the moment. However, we can easily extend the range of supported shapes to other shape primitives, combination of shapes, or even CAD models of the sample, if one is available. However, it was our aim to achieve inspection without having to provide the CAD model. To this end, we also investigated using tactile/force feedback to automatically align the sensor with the surface as proposed in [Moura et al., 2018]. We concluded that in our scenario the friction forces are too high to be used for aligning the tool during motion. This approach would require repeatedly stopping to realign the tool which would significantly increase the time required to complete a scan.

Some of our future work will also focus on scanning more complex shapes such as pipes, joints and tanks surrounded by other structures. This will require us to take collision avoidance into account during motion and tool path planning using methods for avoiding obstacles such as the one we presented in [Yang et al., 2016] and [Yang et al., 2017] and representations of the environment that we proposed in [Ivan et al., 2013] and [Ivan and Vijayakumar, 2015].

## ACKNOWLEDGEMENTS

This research is supported by the EPSRC UK RAI Hub in Offshore Robotics for Certification of Assets (ORCA) (EP/R026173/1) and the UK-India Education and Research Initiative (UKIERI, grant reference UKIERI-DST 2016-17-0152).

## REFERENCES

[Andersen et al., 2013] Andersen, M. S., Dahl, J., and Vandenberghe, L. (2013). Cvxopt: A python package for convex optimization, version 1.1. 6. *Available at cvxopt. org*, 54.

[Carius et al., 2018] Carius, J., Wermelinger, M., Rajasekaran, B., Holtmann, K., and Hutter, M. (2018). Autonomous mission with a mobile manipulator–a solution to the mbzirc. In *Field and Service Robotics*, pages 559–573. Springer.

[Cheeke, 2016] Cheeke, J. D. N. (2016). *Fundamentals and applications of ultrasonic waves*. CRC press.

[Chitta, 2016] Chitta, S. (2016). *MoveIt!: An Introduction*, pages 3–27. Springer International Publishing, Cham.

[Chitta et al., 2017] Chitta, S., Marder-Eppstein, E., Meeussen, W., Pradeep, V., Rodríguez Tsouroukdissian, A., Bohren, J., Coleman, D., Magyar, B., Raiola, G., Lüdtke, M., and Fernández Perdomo, E. (2017). ros_control: A generic and simple control framework for ros. *The Journal of Open Source Software*.

[Chum and Matas, 2005] Chum, O. and Matas, J. (2005). Matching with prosac - progressive sample consensus. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 220–226 vol. 1.

[Clough et al., 2017] Clough, M., Fleming, M., and Dixon, S. (2017). Circumferential guided wave emat system for pipeline screening using shear horizontal ultrasound. *NDT & E International*, 86:20–27.

[Dang et al., 2011] Dang, D., Lamiraux, F., and Laumond, J. (2011). A framework for manipulation and locomotion with realtime footstep replanning. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 676–681.

[Dong Hun Shin et al., 2003] Dong Hun Shin, Hamner, B. S., Singh, S., and Myung Hwangbo (2003). Motion planning for a mobile manipulator with imprecise locomotion. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 847–853 vol.1.

[Fallon et al., 2015] Fallon, M., Kuindersma, S., Karumanchi, S., Antone, M., Schneider, T., Dai, H., D'Arpino, C. P., Deits, R., DiCicco, M., Fourie, D., et al. (2015). An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics*, 32(2):229–254.

[Ferreau et al., 2014] Ferreau, H., Kirches, C., Potschka, A., Bock, H., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363.

[Fischler and Bolles, 1987] Fischler, M. A. and Bolles, R. C. (1987). *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, page 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Gill et al., 2005] Gill, P. E., Murray, W., and Saunders, M. A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*

[Hellier, 2012] Hellier, C. (2012). *Handbook of Nondestructive Evaluation, Second Edition*. McGraw-Hill Education.

[Hirao and Ogi, 1999] Hirao, M. and Ogi, H. (1999). An sh-wave emat technique for gas pipeline inspection. *Ndt & E International*, 32(3):127–132.

[Hirao and Ogi, 2003] Hirao, M. and Ogi, H. (2003). *Emats for Science and Industry: Noncontacting Ultrasonic Measurements*. Kluwer Academic Publishers.

[Hootsmans et al., 1992] Hootsmans, N. A. M. et al. (1992). *The motion control manipulators on mobile vehicles*. PhD thesis, Massachusetts Institute of Technology.

[Ivan and Vijayakumar, 2015] Ivan, V. and Vijayakumar, S. (2015). Space-time area coverage control for robot motion synthesis. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 207–212.

[Ivan et al., 2019] Ivan, V., Yang, Y., Merkt, W., Camilleri, M. P., and Vijayakumar, S. (2019). *EXOTica: An Extensible Optimization Toolset for Prototyping and Benchmarking Motion Planning and Control*, pages 211–240. Springer International Publishing, Cham.

[Ivan et al., 2013] Ivan, V., Zarubin, D., Toussaint, M., Komura, T., and Vijayakumar, S. (2013). Topology-based representations for motion planning and generalization in dynamic environments with interactions. *IJRR*, 32(9-10):1151–1163.

[Kang and Li, 2015] Kang, Z. and Li, Z. (2015). Primitive fitting based on the efficient multibaysac algorithm. *PLOS ONE*, 10(3):1–21.

[Kim et al., 2019] Kim, S., Jang, K., Park, S., Lee, Y., Yup Lee, S., and Park, J. (2019). Whole-body control of non-holonomic mobile manipulator based on hierarchical quadratic programming and continuous task transition. In *IEEE International Conference on Advanced Robotics and Mechatronics (ARM)*.

[Luo and Rose, 2003] Luo, W. and Rose, J. L. (2003). Guided wave thickness measurement with emats. *Insight-Non-Destructive Testing and Condition Monitoring*, 45(11):735–739.

[Merkt et al., 2019a] Merkt, W., Ivan, V., and Vijayakumar, S. (2019a). Continuous-time collision avoidance for trajectory optimization in dynamic environments. In *Submitted, under review for IROS 2019*.

[Merkt et al., 2019b] Merkt, W., Ivan, V., Yang, Y., and Vijayakumar, S. (2019b). Towards shared autonomy applications using whole-body control formulations of locomanipulation. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1206–1211.

[Merkt et al., 2017] Merkt, W., Yang, Y., Stouraitis, T., Mower, C., Fallon, M., and Vijayakumar, S. (2017). Robust Shared Autonomy for Mobile Manipulation with Continuous Scene Monitoring. In *CASE*, pages 130–137. IEEE.

[Meza et al., 2007] Meza, J. C., Oliva, R. A., Hough, P., and Williams, P. (2007). OPT++: An object-oriented toolkit for nonlinear optimization. *ACM Trans. on Math. Softw. (TOMS)*.

[Moore and Stouch, 2014] Moore, T. and Stouch, D. (2014). A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer.

[Moura et al., 2018] Moura, J., Mccoll, W., Taykaldiranian, G., Tomiyama, T., and Erden, M. S. (2018). Automation of train cab front cleaning with a robot manipulator. *IEEE Robotics and Automation Letters*, 3(4):3058–3065.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 3, page 5. Kobe, Japan.

[Ribichini et al., 2012] Ribichini, R., Cegla, F., Nagy, P. B., and Cawley, P. (2012). Evaluation of electromagnetic acoustic transducer performance on steel materials. *NDT&E International*.

[Sharma et al., 2017] Sharma, G., Goyal, R., Liu, D., Kalogerakis, E., and Maji, S. (2017). Csgnet: Neural shape parser for constructive solid geometry. *CoRR*, abs/1712.08290.

[Torr and Zisserman, 2000] Torr, P. and Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138 – 156.

[Tulsiani et al., 2016] Tulsiani, S., Su, H., Guibas, L. J., Efros, A. A., and Malik, J. (2016). Learning shape abstractions by assembling volumetric primitives. *CoRR*, abs/1612.00404.

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

[Wan and Van Der Merwe, 2000] Wan, E. A. and Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158.

[Xin et al., 2018] Xin, G., Lin, H., Smith, J., Cebe, O., and Mistry, M. (2018). A model-based hierarchical controller for legged systems subject to external disturbances. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4375–4382.

[Yamamoto and Yun, 1992] Yamamoto, Y. and Yun, X. (1992). Coordinating locomotion and manipulation of a mobile manipulator. In *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pages 2643–2648 vol.3.

[Yang et al., 2016] Yang, Y., Ivan, V., Merkt, W., and Vijayakumar, S. (2016). Scaling sampling-based motion planning to humanoid robots. In *IEEE ROBIO*.

[Yang et al., 2017] Yang, Y., Merkt, W., Ferrolho, H., Ivan, V., and Vijayakumar, S. (2017). Efficient Humanoid Motion Planning on Uneven Terrain Using Paired Forward-Inverse Dynamic Reachability Maps. *IEEE RA-L*, 2(2):522–529.

[Yang et al., 2018] Yang, Y., Merkt, W., Ivan, V., and Vijayakumar, S. (2018). Planning in time-configuration space for efficient pick-and-place in non-static environments with temporal constraints. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9.

[Zou et al., 2017] Zou, C., Yumer, E., Yang, J., Ceylan, D., and Hoiem, D. (2017). 3d-prnn: Generating shape primitives with recurrent neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 900–909.