Control Framework for a Hybrid-steel Bridge Inspection Robot

Hoang-Dung Bui, Son T. Nguyen, U-H. Billah, Chuong Le, Alireza Tavakkoli, Hung M. La, *IEEE Senior Member*

Abstract— **Autonomous navigation of steel bridge inspection robots are essential for proper maintenance. Current robotic systems are controlled manually through cables or remote commands, and require continuous human involvement. In this paper, we propose a control system framework for our previously designed ara robot [1] as an effort to minimize human involvement. The ara robot can automatically configure itself into two transformations (mobile and worming) through this framework. We integrated a distance control architecture for magnetic adhesion to steel in both untouched and touched modes. In addition, a switching control was incorporated in the robot for processing 3D point clouds of steel surfaces. The surface availability algorithm (considers plane, area and height) of the switching control enables the robot to perform inch-worm jumps autonomously. Moreover, the *Mobile transformation* allows the robot to move on continuous steel surfaces and perform visual inspection. The experimental results of this work represent the impact of switching control for autonomous navigation. Furthermore, we present the preliminary results of visual inspections performed using our proposed robotic system.**

I. Introduction

Environmental degradation, continuous friction with vehicle tires, overloading, and numerous other causes contribute to the deterioration of steel bridge materials. Continuous steel bridge monitoring is necessary to ensure transportation safety and proper maintenance. Most of these bridges are monitored by civil inspectors manually [2]. However, the complex structure (pipes and poles) and unsafe location of steel bridges, make the inspection process a perilous task for human inspectors. Additionally, manual inspection is time consuming, labor-intensive, and disruptive to traffic.

To alleviate these complications associated with manual inspection, a variety of robotic systems were recently developed. These robots were equipped with different adhesion mechanisms [3]–[13], cameras (RGBD, stero), and other sensors (IMU, eddy current). Magnetic adhesion in untouched mode enabled the MaggHD [14] robot to navigate flexibly on smooth steel surfaces. On the other hand, legged robots integrated with electromagnets were designed to move in complex steel structures [15]. These robots are designed for a particular environment, lacking the deploy-ability in many unstructured environments. As a result, a practical tank-like robot design was implemented by [2] using five-degrees of freedom (DOF) arms and eddy current sensors. The arms enabled the robot to perform eddy current measurement flexibly and efficiently on practical steel bridges. Another type of tank-like robot was developed by [16] with untouched magnet blocks to move efficiently on metal surfaces. Furthermore, a climbing robot approach was developed by [2] to check the contoured weld of steel bridges using RGB-D camera and eddy current sensors. Though these robots alleviated the difficulty of moving on complex steel surfaces, they were controlled manually by cables or remote instruction from human operators.

As an effort to navigate robots autonomously, a hybrid robot design (named as ara robot) was proposed by our previous research [1]. This robot can efficiently transit from one steel bar to another as well as contoured surfaces of steel bridges. Since steel bridge inspection is a continuous process, the primary goal of our research is to develop a fully autonomous robotic system to automate this task. Therefore, we divided our research into four main stages: 1) Hybrid steel bridge inspection robot design [1], 2) Control framework for autonomous robot navigation and inspection, 3) 3D SLAM deployment and in-depth inspection and 4) Structural health monitoring map generation.

In this paper, we propose a control framework for autonomous robot navigation on steel bridge structures as part of stage 2 of our research. A switching control mechanism is integrated in the control framework. The switching control allows the robot to transition into two different transformations, i.e., mobile and worming. The switching control determines the availability of the planar surface, its area and height for determining its next transition. We propose an area estimation algorithm from the 3D point cloud data in this work. This algorithm determines if the available area is sufficient enough for the robots foot transition. Based on the height estimation on switching control, the robot chooses its transformation. The robot performs an inch-worm jump when *Worming transformation* is activated. For navigation in *Mobile transformation*, we have proposed a path planning control framework. For performing visual inspection, we integrated an encoder-decoder based convolutional neural network [17] in this module. Moreover, a magnetic array based distance control is proposed in this work for autonomous magnetic adherence to steel surface. The rest of the paper is arranged as follows: in section II we elaborately

This work is supported by the U.S. National Science Foundation (NSF) under grants NSF-CAREER: 1846513 and NSF-PFI-TT: 1919127, and the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OST-R) under Grant No. 69A3551747126 through INSPIRE University Transportation Center.The views, opinions, findings and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NSF and USDOT/OST-R.

The authors are with the Advanced Robotics and Automation (Ara) Lab, Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA. Corresponding author: Hung La, email: hla@unr.edu.

discuss our methodology, while the experimental results of this work are discussed in section IV. Lastly, we conclude in section V.

II. Overall Architecture

The ara robot can configure itself into two transformation mode i.e., mobile and worming. In this work, we integrated a switching control mechanism (shown in Fig. 1) to the ara robot. This control enables the robot to transform its configuration depending on environmental conditions. In continuous and smooth steel surfaces, the robot activates the *Mobile transformation* (shown in Fig. 2(a)). The robot navigates using a path planning control, navigates using differential wheel and performs visual inspection in this configuration. Moreover, the magnetic array is switched to the untouched mode in *Mobile transformation*. The robot configures itself into *Worming transformation* (shown in Fig. 2(b)) when it detects a complex steel surface. The robot performs an inch-worm jump to the next surface since it cannot move on wheels on complex structures. The magnetic array switches into touched mode while performing the jump. The switching control mechanism controls the movement of the robot, detects environment type and sends the appropriate command to executable nodes.

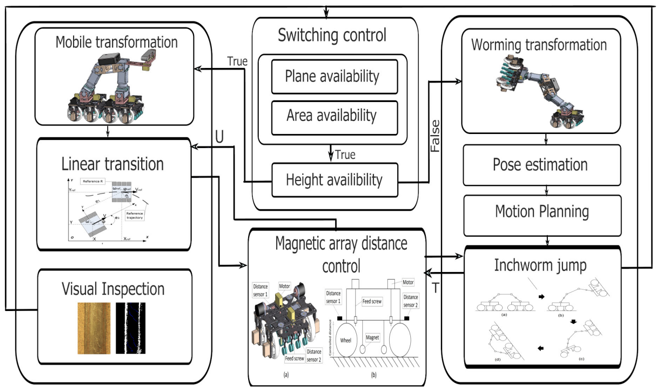


Fig. 1: The proposed control system framework for autonomous navigation

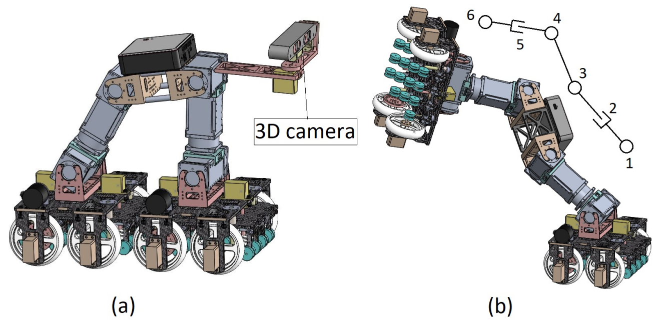


Fig. 2: Steel bridge inspection robot in (a) mobile transformation and (b) *Worming transformation*

The control architecture of the ara robot is comprised of multiple low-level and high-level control structures. Several tasks are performed by the low-level control structure (Arduino). The velocity conversion is performed by this control for robot wheel movement. Additionally, the function of the magnetic array is controlled by this low-level control. Furthermore, it reads the encoder and Inertial Measurement Unit (IMU) sensor measurements and passes this information to the high-level control. We embedded the high-level control in an on-board processor (NUC). The NUC processor enables the switching control function, processing of 3D point cloud data, and the inverse kinematics of the *Worming transformation*. The sensor data is fused by the high-level and low-level controls using a Kalman filter. This data is utilized for achieving the desired linear velocity and data acquisition from advanced sensors (RGB camera, IMU). The arrangement of the high level and low-level controls is shown in Fig. 3.

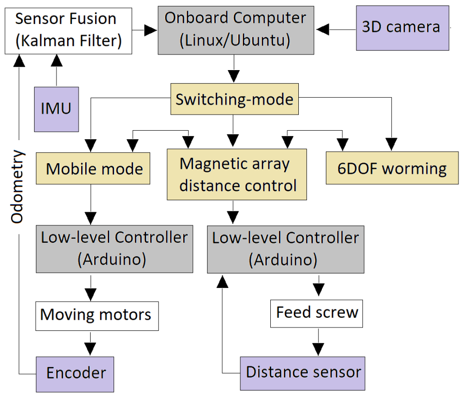

Sensor Fusion (Kalman Filter)
Onboard Computer (Linux/Ubuntu)
3D camera
IMU
Switching-mode
Odometry
Mobile mode
Magnetic array distance control
6DOF worming
Low-level Controller (Arduino)
Low-level Controller (Arduino)
Moving motors
Feed screw
Encoder
Distance sensor

Fig. 3: The control architecture integrated into the ara robot [1]

III. Control System Framework

The control system framework of the ara robot is composed of four modules: switching control, *Mobile transformation*, *Worming transformation* and magnetic array distance control. An overview of the overall framework is shown in Fig. 1.

A. Switching Control

The switching control S enables the robot to autonomously configure itself into two transformations (mobile and worming). The control employs switching function S, represented in Eq.1. The function takes as input three boolean parameters: plane availability S_{pa}, area availability S_{am} and height availability S_{hc}. These parameters determine if there is any still surface available, while enabling the estimation of the area of the surface and its height. A logical operation is performed on these parameters using function $f(.)$. The function parameters are estimated from 3D point cloud data of steel surface.

$$S = f(S_{pa}, S_{am}, S_{hc}) = S_{pa}S_{am}S_{hc}. \quad (1)$$

The robot configures to *Mobile transformation* if the function return a true value. The false value configures the robot into the *Worming transformation*.

Plane availability: The 3D point cloud of steel surface is processed using *pass-through* filtering, *downsampling*, and

Manuscript 1976 submitted to 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2020.

plane detection [18]. The *plane detection* [18] extracts the planar point cloud P_{cl} from the initial point cloud. The plane availability is checked using Eq.2:

$$\begin{cases} S_{pa} = False,\, if\, P_{cl} = \emptyset \\ S_{pa} = True,\, otherwise \end{cases} \tag{2}$$

Moreover, two functions, *get_centroid* and *get_normal_vector*, use the point cloud set to calculate the point cloud's centroid $C_{P_{cl}}$ and normal vector $\vec{N_{P_{cl}}}$.

Area availability: The robot legs require an area estimation of the available planar surface area P_{cl}. It is essential to ensure the availability of sufficient area for successful robot transition. For this purpose we proposed **Algorithm 2** to check the sufficiency of the available planar surface. However, the most essential component of this area estimation algorithm is to extract the boundary points of the planar surface. Hence, a boundary estimation procedure is also proposed in **Algorithm 1**.

Algorithm 1 Boundary point estimation from 3D point cloud data of steel bridges

```
 1: procedure BoundaryEstimation($P_{cl}, \alpha_s$)
 2:     $Planes = \{xy, yz, zx\}$
 3:     $d_{min}$ = $\forall_{i \in Planes}$ Point along minimum value of plane $i$
 4:     $d_{max}$ = $\forall_{i \in Planes}$ Point along maximum value of plane $i$
 5:     Initialize $B_s = \{\}$
 6:     for $p \in Planes$ do
 7:         $i \rightarrow 1$
 8:         while $sl_{p_i} < d_{max}$ do
 9:             $sl_{p_i} = d_{min_p} + i * \alpha_s$
10:             $PS_{p_i}$ = Set of points in range $sl_{p_i} \pm \alpha_s/2$
11:             $P_{cl_A}, P_{cl_B} = \underset{\forall\{P_i,P_j\} \in PS_{p_i}}{\text{argmax}} \{\|P_i - P_j\|\}$
12:             $B_s = B_s \cup \{P_{cl_A}, P_{cl_B}\}$
13:             $i = i + 1$
14:         end while
15:     end for
16: return $B_s$
17: end procedure
```

The boundary points in **Algorithm 1** are estimated by a window-based approach. The input of the algorithm is the 3D point cloud P_{cl} of the estimated planar surface and a slicing parameter α_s. At first, we calculate the two furthest points represented as d_{min} and d_{max} in the point cloud P_{cl} along each plane. Then the point cloud is divided into multiple smaller slices along the three planes. For each slice in a particular plane p we calculate the slicing index sl_p, which represents the center coordinate of the slice as shown in line 8 of algorithm 1. After that, the point sets PS_p in the range $sl_p \pm \alpha_s/2$ is extracted from P_{cl}. This sliding factor is experimentally determined based on the point cloud size. For each set of points from PS_p, we extract the two furthest points (P_{cl_A}, P_{cl_B}). These two points are estimated as the boundary point for that particular slice and added to the boundary point sets B_s. A pictorial representation of the boundary estimation algorithm is shown in Fig.4.

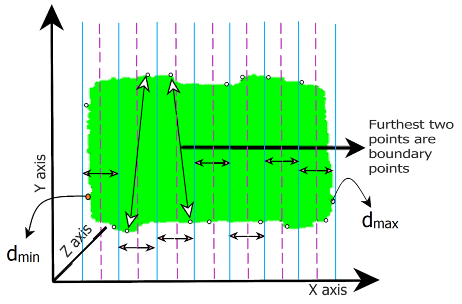


Fig. 4: Boundary point estimation from 3D point cloud data

After estimating the boundary points B_s, we estimate the area availability parameter S_{am}. The estimation is performed using the **Algorithm 2**. The input of this algorithm are the boundary points B_s, point cloud centroid $C_{P_{cl}}$, normal vector of point cloud $\vec{n_{P_{cl}}}$, length l and width w of robot leg and wheel distance tolerance t. At first we calculate the n closest points (N_{clos}) from B_s to the point cloud centroid $C_{P_{cl}}$. For each point, N_i in the set N_{clos}, a set of computations is performed to estimate the plane corners for adherence to robot wheels. At first, the coordinate frame vectors $\vec{e_{x_i}}, \vec{e_{y_i}}$ and $\vec{e_{z_i}}$ are calculated for point N_i. In the next step, the algorithm estimates the corner points of a rectangle of width w and length l, which is also robot foot width and length respectively. We estimate the rectangle edges parallel along the vectors $\vec{e_{x_i}}$ and $\vec{e_{y_i}}$. Therefore, the four corners R of the rectangle are estimated using these two vectors. Additionally, we include four middle points of the estimated rectangle corners in R to alleviate point cloud collection error as well as accommodate for the non-convex shape of the steel surface. After the estimation step, we find m closest points to R from the B_s to measure if the points in R are inside the boundary. Hence, we calculate the distance from point cloud centroid $C_{P_{cl}}$ to R and Q respectively. The algorithm considers a point lies inside the boundary if its tolerance is less than t and the distance to centroid should be less than its neighbors. When the value of S_{am} is true for all the conditions, we consider those sets of points as rectangular points.

Height availability: The height availability S_{hc} is crucial for the switching control. Based on this parameter, the switching control activates the robot transformations. At first the centroid of the point cloud $C_{P_{cl}}$ is calculated along the camera frame f_c. Then we transform the centroid coordinate to the robot base frame f_{rb} using Eq. 3.

$$P_{C_{f_{rb}}} = T_{f_{rb}f_c} P_{C_{f_c}} \tag{3}$$

where $p_{C_{f_{rb}}}$, $p_{C_{f_c}}$ is coordinate of the centroid C_c in the camera frame and the robot base frame, respectively. $T_{f_{rb}f_c}$ is the transformation matrix from the camera frame f_c to the robot base frame f_{rb}.

The plane height $z_{f_{rb}}$ coordinate is then compared with the robot base height. If they are equal, the returned result is *true*

Algorithm 2 Area Checking from the plane surface boundary points and Pose Calculation

```
1:  procedure AREA($B_s, C_{P_{cl}}, \vec{n_{P_{cl}}}, w, l, t, S_{am}$)
2:      $N_{clos}$ = Find $n$ closest points to $C_{P_{cl}}$ from $B_s$
3:      for $N_i \in N_{clos}$ do
4:          $R = \{\}$, //Estimated rectangle corner points,
5:          $\vec{e_{x_i}} = N_i - C_{P_{cl}}$
6:          $\vec{e_{z_i}} = \vec{n_{P_{cl}}}$
7:          $\vec{e_{y_i}} = \vec{e_{x_i}} \times \vec{e_{y_i}}$
8:          $k_w = \frac{w}{|\vec{e_{x_i}}|} e_{y_i}$ and $k_l = \frac{b}{|\vec{e_{y_i}}|} e_{x_i}$,
9:          $\{R_1, R_2\} = \{N_i + k_w, N_i - k_w\}$
10:         $R = R \cup \{R_1, R_2\}$
11:         $R = R \cup \{R_1 + k_l, R_2 + k_l\}$
12:         $M = \forall_{r_i \in R} \{\frac{r_i + r_{i+1}}{2}\}$
13:         $R = R \cup M$
14:         $S_{am} = True$
15:         for $r_i \in R$ do
16:             $Q_i$ = Find $m$ closest points to $r_i$
17:             $d_{r_i} = \|d_{r_i}, C_{P_{cl}}\|$ and $d_{Q_i} = \|Q_i, C_{P_{cl}}\|$
18:             $S_i = (d_{r_i} < d_{Q_i}) \vee (\frac{d_{r_i} - d_{Q_i}}{d_{r_i}} < t$
19:             $S_{am} = S_{am} \wedge S_i$
20:         end for
21:         Pose = {orientation, Position}
22:         if $S_{am}$ == True then
23:             $R_c$ = Centroid of $R$
24:             orientation = $(\vec{e_{x_i}}, \vec{e_{y_i}}, \vec{e_{z_i}})$
25:             Position = $(x_{R_c}, y_{R_c} - l/4, z_{R_c})$
26:             return Pose
27:         end if
28:     end for
29:     return False
30: end procedure
```

and the robot configure to *Mobile transformation*. Otherwise, it returns *false* and the robot go to *Worming transformation*. The height availability condition is shown as in Eq.4.

$$\begin{cases} S_{hc} = True,\ if\ z_{f_{rb}} = z_{robotbase} \\ S_{hc} = False,\ otherwise \end{cases} \tag{4}$$

B. Magnetic Array Distance Control

The magnetic array of distance control switch the magnetic array of the ara robot into two modes such as touched and untouched. The distance control framework is shown precisely in Fig.5. The *Mobile transformation* configures the magnetic array into untouched mode using this framework. This mode creates a magnetic adhesion force by keeping a 1mm distance from the steel surface. On the other hand, the distance control framework switches the magnetic array to touched mode, when *Worming transformation* occurs. The touched mode allows the robot to fully adhere to the surface for performing an inch-worm jump.

The adhesion force of the magnetic array is manipulated with a PID controller as shown in Fig.5. This controller enables the magnetic array to move up and down for adjusting magnetic adhesion force to the steel surface. Two high-resolution distance sensors (VL6180X) are mounted on the magnetic array for distance sensing. For transmitting the distance to the controller, two motors and a parallel screw was incorporated in the controller. Since the magnetic array needs to maintain a 1mm distance to the steel surface, the accuracy of the controller is crucial. A minor mechanical error may lead to part of the magnetic array transform into the touching mode and result in a huge load to the two motors. Therefore, this design improves the accuracy of the controller significantly to avoid the aforementioned difficulties.

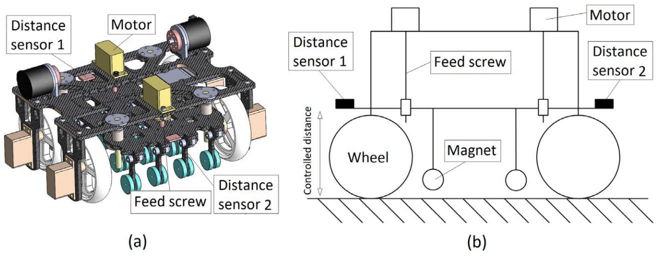


Fig. 5: Distance control system of magnetic array; a) 3D model b) 2D diagram.

C. Mobile transformation

The ara robot switches to *Mobile transformation* to move on the continuous steel surface. The transition in the smooth surface is performed by the two-wheel of each robot leg. To perform this navigation smoothly at first we defined the problem statement of two-wheel movement in Fig. 6.

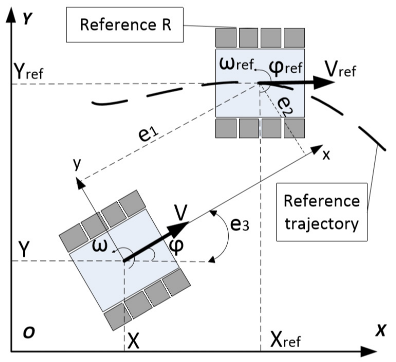


Fig. 6: Statement of the problem of robot trajectory movement control.

The dynamic equation of the robot (Eq.5) is derived from Fig.6 [8]. The current position and the orientation of the robot is represented with (x_c, y_c, φ_c).

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\varphi}_c \end{bmatrix} = \begin{bmatrix} \cos\varphi_c & 0 \\ \sin\varphi_c & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} \tag{5}$$

The robot needs to move a target position $R(x_r, y_r)$ with constant velocity v_r with the heading angle ϕ_r. The reference path coordinates satisfy Eq.6.

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\varphi}_r \end{bmatrix} = \begin{bmatrix} \cos\varphi_r & 0 \\ \sin\varphi_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{6}$$

To navigate the robot to point R, a control is designed to help the robot to track the point R. A tracking error vector is introduced $e = [e_1, e_2, e_3]^T$, presented as in Eq. (7).

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\varphi_c & \sin\varphi_c & 0 \\ -\sin\varphi c & \cos\varphi_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x_c \\ y_r - y_c \\ \varphi_r - \varphi_c \end{bmatrix} \quad (7)$$

Derivative of Eq.(7) and linearizing using [8], we get Eq.8 & 9:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} e_2\omega_c - v_c + v_r \cos e_3 \\ -e_1\omega_c + v_r \sin e_3 \\ \omega_r - \omega_c \end{bmatrix} \quad (8)$$

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} v_c + \begin{bmatrix} e_2 \\ -e_1 \\ -1 \end{bmatrix} + \begin{bmatrix} v_r \cos e_1 \\ v_r \sin e_2 \\ \omega_r \end{bmatrix} \quad (9)$$

To track the reference point R from the current position C, the error vector need to go to 0: $e_i \rightarrow 0$. A mixed PID controller design was integrated into the ara robot for this purpose. The design of the mixed PID controller for path planning is shown in Fig.7. The trajectory is divided into discrete tracking positions as a set point of the first loop controller. The distance error is derived to go to zero by this loop. The second loop handles the robot heading error obtained by $atan(y_r - y_c, x_r - x_c)$. A mixer combines outputs of two controllers then provides the velocity to motors. The feedback of position and orientation is gotten by encoders and IMU sensors.

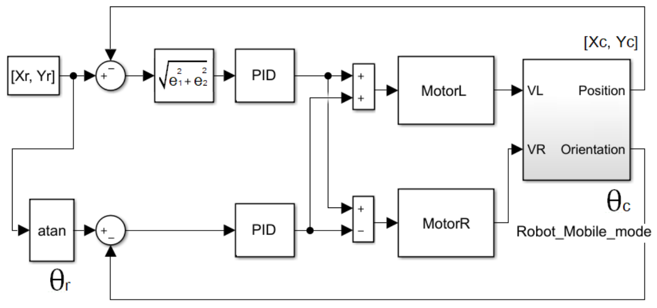


Fig. 7: The mixed PID controller for path planning in mobile mode.

The inspection framework: After performing the linear transition, the robot conducts the visual inspection. The on-board RGB camera module captures images from the steel surface and sends them to the NUC processor for defect identification. For steel bridge defect identification, we have used an encoder-decoder based CNN [17]. This architecture is inspired by our work in detecting defects on the surface of concrete structures [19], [20]. The CNN architecture segments images into defect and healthy regions. The input images are passed through five encoder layers followed by five decoder layers. Each encoder performs a 7×7 convolution operation, to extract defect feature maps. The convolution operation is followed by a batch normalization operation and ReLU activation. The feature space is down-sampled using a 2×2 max-pooling unit and fed to the next encoder layer. We utilize five such encoder layers for feature extraction. Once the last encoder layer is reached, the resultant feature maps enter the decoder portion of the network, where they are up-sampled using bi-linear interpolation in each decoder layer. The up-sampling operations are again followed by a convolution operation, batch normalization, and ReLU activation. A graphical representation of the CNN architecture is shown in Fig.8. We pre-trained the CNN architecture on 3000 steel images containing severe defects (e.g., corrosion). For the hyperparameter optimization, we used the adam optimizer with a learning rate of 0.0001. The network was trained for 150 epochs in this experiment.

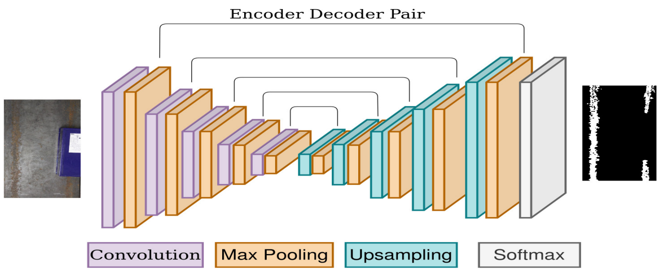


Fig. 8: An encoder decoder based CNN architecture for steel bridge inspection.

D. *Worming transformation*

The worming mode enables the robot to perform an inch-worm jump from one steel surface to another as shown in Fig.9. At first, the second permanent magnet on the second leg of the robot is activated for steel adherence. As a result, a strong adhesive force is generated for the robot stand and perform the worming. A controller manipulates the joints of the arm to move the first leg towards the target plane using generated trajectory. This step is shown in Fig.9(b). When the first leg touches the target surface, the second one starts to detach from the starting surface (shown in Fig.9(c)). Finally in Fig.9(d) the second leg detaches itself from the starting surface and adhere to the target surface.

Converting from Mobile (Fig.2(a)) to *Worming transformation* is challenging for the motion planner to create a trajectory. To have a better performance, a convenient robot pose P_{conv} is proposed where the robot should move to firstly as starting of *Worming transformation*. In Fig.10-left, the pose P_{conv} is in blue color. From there, the motion planner will generate a trajectory to the destination in red color. Moreover, an Inverse Kinematics (IK) solver processes the trajectory and generate a set of joint positions for the robot to move. The worming is completed by moving the second leg to the target surface and reform the Mobile configuration as shown in Fig.10(b). The first step and third move of the robot is done by point-to-point control. To perform the second step, the robot needs to determine the target plane and its pose which are an output of **Algorithm 2**.

The flexibility of the robot in worming is made of the six DoF arm. The revolute joints of the arm in Fig.2(b) can rotate along three different axes separately. For example, the joint 2

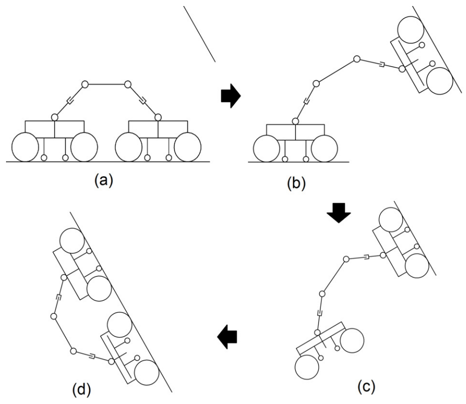


Fig. 9: Inch-worm jump from one steel surface to another.

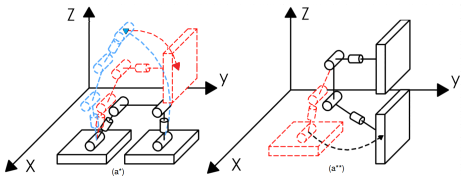


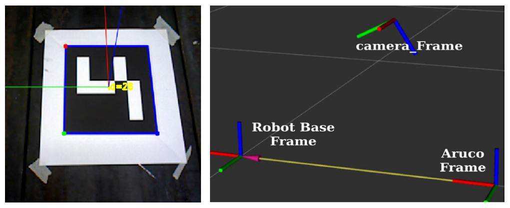
Fig. 10: Inch-worm jump procedure.

and joint 5 in Fig.2(b) are configured to rotate around *y-axis*. The rest of the joints are positioned to rotate around *z-axis*. This configuration was selected for maintaining symmetry so that the manipulator can move efficiently in both worming and mobile mode. Our previous research states an in detail elaboration of the robotic arm in [1].

IV. Experiment Results

In this paper, we experimented on ara robot version 1.0 which derived from [1] with an additional camera module. An RGB-D camera (ASUS Xtion Pro Live) is attached to the robot for point cloud collection and visual inspection. For camera calibration, we integrated the parameters from the method implemented by [18]. We localized the robot using aruco marker [21] for our experiments. The aruco marker was placed on a planar surface, where the robot base can adhere. We performed a geometric calculation to locate the position between aruco marker and the robot base. Additionally, we incorporated an Intel NUC 5 Business Kit (NUC5i5MYHE) - Core i5 vPro for employing the robotic operating system (ROS) as well as the inspection framework. Since robot feet enclose a rectangular area by design, we have drawn a rectangle around robot feet. The robot was configured to its *Mobile transformation* for moving a distance d_c (measured by robots encoder) along the x axis. We place the marker in the center of the rectangle, we have drawn before to achieve equivalent orientation. The pose and position between camera and robot base was extracted using the function *lookupTransform* in *tf* package. The aruco marker and the required coordinates are shown in Fig.11.

We perform our experiment on two individual steel slabs located perpendicularly from each other. The steel slabs are highly corroded to replicate steel defects. The following section describe the experiment and their results elaborately.



(a) (b)

Fig. 11: Aruco Marker & coordinate frames for localization.

A. Switching control

At starting, the RGB-D point cloud data of a bridge steel bar was collected from the robot camera. An example of the initial point cloud is shown in Fig.12(a). After performing some pre-processing operations such as *pass-through filtering* and *downsampling*, the data is sent to *plane detection* to extract the planar surface. The processed point cloud is shown in Fig.12(b). The coordinate frame is also shown in the figure with x-axis in red, y-axis in green, and z-axis in blue. After obtaining the planar surface we extract the

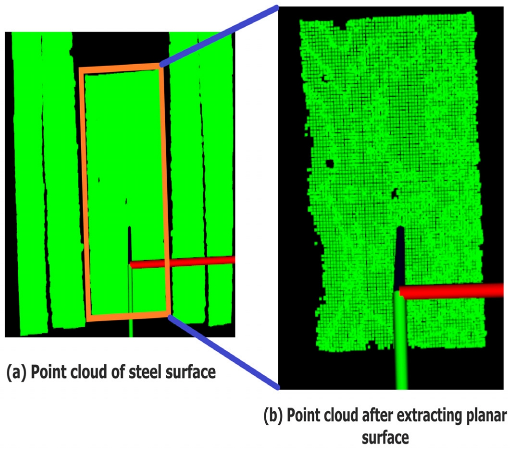

(a) Point cloud of steel surface
(b) Point cloud after extracting planar surface

Fig. 12: Planar surface extraction from 3D point cloud of steel surface.

boundary points of the surface and perform *Area availability* checking. For this purpose, **Algorithm 1** and **Algorithm 2** were employed on two different surfaces, one containing sufficient area for movement and the other without. Using **Algorithm 1**, we have the two boundary points of the two different point cloud as shown in Fig.13(a) and Fig.13(c). The area availability **Algorithm 2** is employed using the boundary point estimated. The parameters of this algorithm are as following : $n = 5, m = 3$ and $t = 0.02$. We estimate five rectangles for robot feet with this algorithm shown in Fig.13(b) and Fig. 13(d) in red, yellow, blue, green, and purple color. In Fig. 13(b), several corners of all the red, yellow, purple, and blue rectangles are outside of the point cloud area. It represents that these rectangles area is not sufficient enough for an inch-worm jump. Only the green rectangle is inside the point cloud, satisfying area requirement. The selected rectangle is shown in Fig. 14(a) (in red color). Since there is enough area for an inch-worm

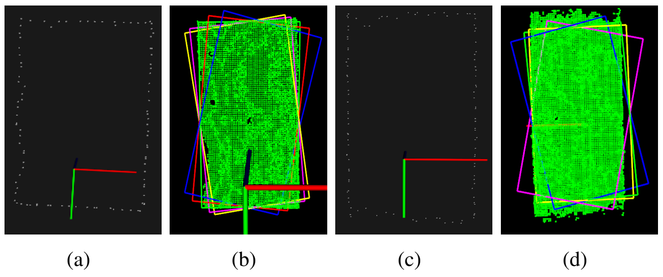


Fig. 13: (a), (c) The Boundary set & (b),(d) Area rectangle set

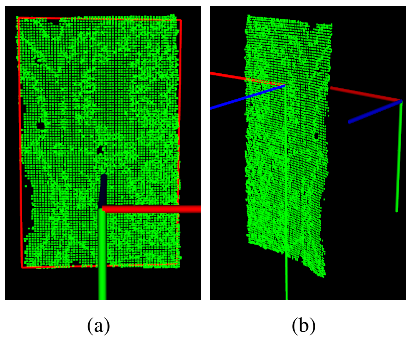


Fig. 14: (a) The selected area rectangle and (b) Pose estimation.

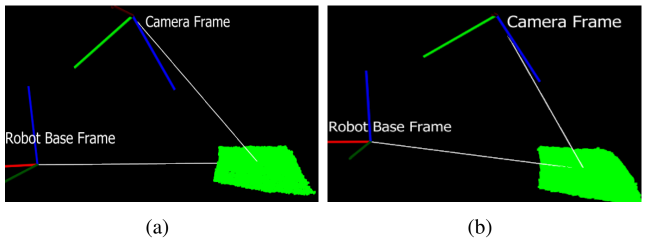


Fig. 15: Surface Height Availability Check (a) Same Height & (b) Different Height.

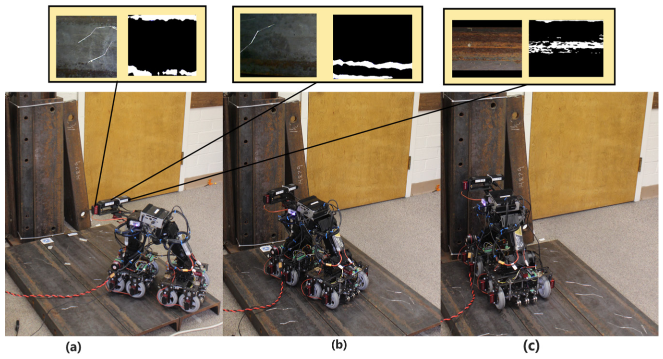


Fig. 16: The movement of robot in *Mobile transformation* and visual inspection.

jump the variable S_{am} is set to true by the algorithm. After that, we calculate the pose of the planar surface in Fig. 14(b). The pose is represented with three orientations shown in red, green, and blue color on the point cloud surface. In Fig. 13(d) the steel surface area is not sufficient for an inch-worm jump. As a result, all the rectangle corners are outside the point cloud boundary.

The planar surface height is assessed after checking area availability. We employed the height availability in two surfaces of different height. Fig. 15(a) displays the point cloud of the surface which the robot is standing on. The camera frame and robot base frame coordinates are shown in Fig. 15(a). We calculated the point cloud centroid with respect to the camera frame. After that, it is transformed into the robot base frame. We compare the plane height (z- coordinate of the centroid of the robot base frame) is compared to robot base height. When both the heights are equal the value of S_{hc} = *True*. The robot will configure itself into *Mobile transformation* in this case.

Fig. 15(b) shows another scenario as the point cloud is from a surface which is $d = 7cm$ lower than the robot base. In this case, the heights are different, then the returned value of variable S_{hc} is *false*, and robot will do the *Worming transformation* in the next step.

B. *Robot navigation in mobile and Worming transformation*

In *Mobile transformation* the robot can move both in x- and y- directions simultaneously. The *Mobile transformation* navigation is represented in Fig. 16.

In *Mobile transformation*, *ara* robot collects steel surface images and performs visual inspection. For this purpose, the steel images are sent to a CNN network. The network segments the image into corroded or healthy regions. A snapshot of the result extracted from the CNN network is shown in Fig. 16(a).

For *Worming transformation*: The result of motion planning for an inch-worm jump from point P_{conv} to the target plane is shown in Fig. 17. We used MoveIt package in ROS with a plane pose as input to generate a trajectory for the first robot foot to reach the destination. To do that, we built a primitive model of the robot in *urdf* format, with the exact dimensions, joint types and limits to *ara* robot. The *Kinematics and Dynamics Library* (KDL) IK and *RRTConnect* motion planner are selected to implement the task. The trajectory - a ROS topic - is a set of robot joint positions, which the robot joint need to follow to reach the target pose. In Fig. 17(a), the robot base frame - *base_link* and the target pose *target_pose_check* are shown. The motion planning needs to create a path for the robot model to go from the initial pose to the target pose. As getting the trajectory, the robot makes a move and goes to the target pose as shown in Fig. 17(b).

After getting the data from the trajectory topic, we send one by one an array of robot joint positions to *ara* robot controller by ROS service. The worming performance of the robot is shown in Fig. 18. In the beginning, the robot activates and lows down the magnetic array to touch the steel surface; then it transforms from the mobile configuration to the convenient point P_{conv} by following a predefined trajectory as shown in Fig. 18(a) and Fig. 18b. As reaching point P_{conv}, the robot starts following the trajectory created by RRTConnect motion planner. As touching the target

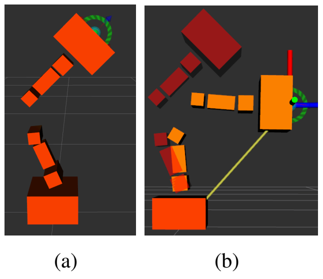


Fig. 17: The motion planning for ara robot movement

surface, it activates both magnetic arrays, the one in the first foot moves down to stick on the target steel surface. Contrarily, the one at the second leg is moving up and detach with the starting surface as shown in Fig. 18(c)-(d). Next, the second foot of the robot will transform into the target plane as shown in Fig. 18(e)-(f).

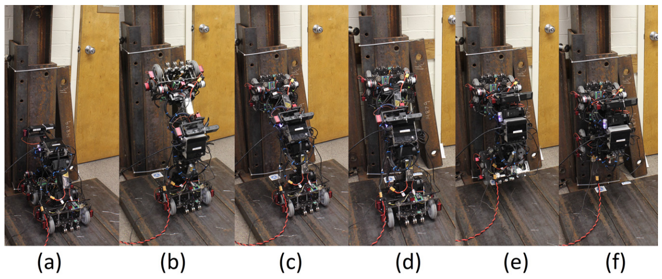


Fig. 18: *Worming transformation*: a) magnetic array of second foot touches the base surface b) first foot moves to convenient point c) first foot reach target pose and touches the second surface d) magnetic array of second foot is released e) and f) second foot moves to target pose

V. Conclusion

A switching control mechanism for autonomous navigation of bridge-inspection robots is proposed in this work. The unique feature of switching in two modes enhances the flexibility of navigation and inspection. The most significant part of this research is the estimation of available surface for navigation using area, plane and height availability. Moreover, the mobile control framework and magnetic adherence distance controller proposed in this work are very important for robot navigation. Nonetheless, the integration of different parts of the framework into the real-world environment was the most challenging part of this research. Further investigation of deployment in actual steel bridges is necessary as the second phase of stage 2 of our research. Therefore, in the future, we intend to investigate the parameters of actual steel bridges to employ autonomous control in the real bridges.

References

[1] S. T. Nguyen, A. Q. Pham, C. Motley, and H. M. La. A practical climbing robot for steel bridge inspection. *IEEE International Conf. on Robotics and Automation*, 2020.

[2] T. S. Nguyen and H. M. La. Development of a steel bridge climbing robot. In *Intelligent Robots and Systems, 2019. IROS 2019. IEEE/RSJ Intern. Conf. on*, Nov 2019.

[3] Anh Q. Pham, Hung M. La, Kien T. La, and Minh T. Nguyen. A magnetic wheeled robot for steel bridge inspection. In Kai-Uwe Sattler, Duy Cuong Nguyen, Ngoc Pi Vu, Banh Tien Long, and Horst Puta, editors, *Advances in Engineering Research and Application*, pages 11–17, Cham, 2020. Springer International Publishing.

[4] H. M. La, T. H. Dinh, N. H. Pham, Q. P. Ha, and A. Q. Pham. Automated robotic monitoring and inspection of steel structures and bridges. *Robotica*, 37(5):947 – 967, May 2019.

[5] Weimin Shen, Jason Gu, and Yanjun Shen. Permanent magnetic system design for the wall-climbing robot. *Applied Bionics and Biomechanics*, 3(3):151–159, 2006.

[6] G. Lee, G. Wu, J. Kim, and T. Seo. High-payload climbing and transitioning by compliant locomotion with magnetic adhesion. *Robotics and Autonomous Systems*, 60(10):1308 – 1316, 2012.

[7] H. Wang and A. Yamamoto. Analyses and solutions for the buckling of thin and flexible electrostatic inchworm climbing robots. *IEEE Transactions on Robotics*, 33(4):889–900, Aug 2017.

[8] J. Guo, W. Liu, and K. M. Lee. Design of flexonic mobile node using 3d compliant beam for smooth manipulation and structural obstacle avoidance. In *2014 IEEE Intern. Conf. on Robotics and Automation (ICRA)*, pages 5127–5132, May 2014.

[9] S. Kamdar. Design and manufacturing of a mecanum sheel for the magnetic climbing robot. *Master Thesis, Embry-Riddle Aeronautical University*, May 2015.

[10] N. H. Pham and H. M. La. Design and implementation of an autonomous robot for steel bridge inspection. In *54th Allerton Conf. on Comm., Con., and Comp.*, pages 556–562, Sept 2016.

[11] N. H. Pham, H. M. La, Q. P. Ha, S. N. Dang, A. H. Vo, and Q. H. Dinh. Visual and 3d mapping for steel bridge inspection using a climbing robot. In *The 33rd Intern. Symposium on Automation and Robotics in Construction and Mining (ISARC)*, pages 1–8, July 2016.

[12] T. Seo and M. Sitti. Tank-like module-based climbing robot using passive compliant joints. *IEEE/ASME Transactions on Mechatronics*, 18(1):397–408, Feb 2013.

[13] D. Zhu, J. Guo, C. Cho, Y. Wang, and K. Lee. Wireless mobile sensor network for the system identification of a space frame bridge. *IEEE/ASME Trans. on Mechatronics*, 17(3):499–507, June 2012.

[14] Magghd. `https://eddyfi.com/en/product/magghd/`.

[15] T. Bandyopadhyay, R. Steindl, F. Talbot, N. Kottege, R. Dungavell, B. Wood, J. Barker, K. Hoehn, and A. Elfes. Magneto: A versatile multi-limbed inspection robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2253–2260, Oct 2018.

[16] Versatrax 100^{TM}. `http://inuktun.com/en/products/`.

[17] Billah Umme Hafsa, La Hung Manh, and Tavakkoli Alireza. Deep learning based feature silencing for accurate concrete crack detection. *Automation in Construction*, page Under Review, 2019.

[18] Hoang-Dung Bui, Hai Nguyen, La Hung Manh, and Li Shuai. A deep learning-based autonomous robot manipulator for sorting application. In *2020 IEEE International Conference on Robotics Computing*, page accepted. IEEE, 2020.

[19] Umme Hafsa Billah, Alireza Tavakkoli, and Hung Manh La. Concrete crack pixel classification using an encoder decoder based deep learning architecture. In *International Symposium on Visual Computing*, pages 593–604. Springer, 2019.

[20] Umme Hafsa Billah, Hung Manh La, Alireza Tavakkoli, and N Gucunski. Classification of concrete crack using deep residual network. In *Proceedings of the 9th International Conference on Structural Health Monitoring of Intelligent Infrastructure (SHMII-9), St. Louis, MO, USA*, pages 4–7, 2019.

[21] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.