Navigation Framework for a Hybrid Steel Bridge Inspection Robot

Hoang-Dung Bui, Hung M. La, IEEE Senior Member

Abstract—Autonomous navigation is essential for steel bridge inspection robot to monitor and maintain the working condition of steel bridge. Majority of existing robotic solutions requires human support to navigate the robot doing the inspection. In this paper, a navigation framework is proposed for ARA robot [1], [2] to run on mobile mode. In this mode, the robot needs to cross and inspect all the available steel bars. The most significant contributions of this research are four algorithms, which can process the depth data, segment it into clusters, estimate the boundaries, construct a graph to represent the structure, generate a shortest inspection path with any starting and ending points, and determine available robot configuration for path planning. Experiments on steel bridge structures setup highlight the effective performance of the algorithms, and the potential to apply to the ARA robot to run on real bridge structures. We released our source code in Github for the research community to use.

I. INTRODUCTION

In the field of steel bridge structure monitoring, there is a significant attention of developing novel robots in the recent years. The influence of environment and transportation such as rain, solar radiation, overloading, friction degrade the bridge continuously. Thus, monitoring and maintenance are essential to ensure the operation safety. The tasks can be done manually, however, it is time consuming, labor intensive, dangerous, affect to the traffic, and sometimes inaccessible for human in complex structures. For the reasons, there are varieties of robotic platforms [1], [3]–[6] developed to support human to do the task. These robots are magnetic-based that help them traverse on multiple angles of steel bridge structures. Most of the robots are controlled manually by an operator.

As an effort to go further in the field, we are developing a bio-inspired hybrid robot - ARA robot [1], [2] (Fig. 1) with the aim to inspect the steel bridge structure autonomously. The robot is able to work in two modes: (1) mobile to traverse on smooth steel surface, and (2) inchworm - to change/jump to another steel bar surfaces. To continue the work of design and dynamic analysis [1] and control [2], in this paper a navigation framework is proposed for the robot to move easily and autonomously on smooth steel surface.

Fig. 1. ARA robot model in (a) mobile, (b) inch-worm modes, and (c) real robot

Navigation is a critical function for robot’s operation in an environment, particularly for autonomous steel bridge inspection robot. The methods in [7]–[9] help the robots move in a local area, and assume the robot dimension is quite small comparing to the workspace. The limit of steel bar dimension and variety of steel bridge structure in our research require a new approach for the ARA robot to handle the navigation and motion planning tasks.

There are variety of structures in steel bridge as shown in Fig. 2 which consists of popular structures as Cross-, T-, I-, K- and L- shape. The structure complexity and dimension make motion planning task is very challenging, requires the perception of robot about the structure and a method that is able to make use of limited work space. Moreover, to navigate the robot inspect on bridge continuously, the navigation system need to build a path in large scale. Combining with a depth sensor to collect data in a far distance, we develop a method to estimate a graph, which represents the steel bridges as a set of nodes and edges. Moreover, we propose a variant of open Chinese Postman Problem (VOCPP), which determines the shortest path to inspect all available steel bars on the structure with difference of starting and ending points.

II. RELATED WORK

There is large number of work related to the navigation of inspection robot for steel bridges [7]–[9]. In [7] particularly for autonomous steel bridge inspection robot, the authors proposed a task-level primitive and online navigation combining with IR sensor, which helps the robot move in local area. In [8], the authors proposed a method to detect edges on a large surface, which is used in navigation. The method in [9] supported the small size robot to move in a large-inside steel bridge space. Their navigation work here assumed that the robot dimension is quite small comparing to the workspace. With the limit of steel bar dimension,
our research needs to handle a new circumstance for robot navigation and motion planning.

There is an important feature in navigation for steel bridge inspection robots: the dimension of steel bars are limited, and the robots have small space to make a motion. The methods in [10]–[12] are able to build very nice convex regions, which make the construction of configuration space easy. The approximation methods, however, reduce the dimension of the workspace, and could make the robot motion infeasible. To overcome these problems, in this paper we propose a method, which segments the workspace into multiple clusters and represent it by a set of boundary points. The method can use all the possible area in the workspace, thus increase the probability of finding a path for the robot. With the irregular shape of the steel bar, Expectation Maximization - Gaussian Mixture Model (EM-GMM) method [13]–[16] is utilized to segment the data into irregular dimensional clusters.

When perceiving the working space, the navigation system represents the bridge structure as a graph. Estimating features from point cloud data get a significant attention [17]–[19]. Those researches worked on a particular small object to the view space of the depth sensors. In our research, the bridge is large and usually over than the sensor view. A graph construction algorithm is developed for this purpose.

From the graph, the next step is to determine a shortest path to inspect all steel bars in the structure, that is called inspection route or Chinese Postman Problem (CPP). There are several requirements for the shortest path in our context. The real bridge is too long for the depth sensor to collect data in one frame, thus the bridge is separated into multiple parts. As the robot finishes inspection in one part, it will move the the next one to continue the task. Therefore, the robot starts at one point and ends in another point. The starting and ending points can be any points in the bridge structure, which are convenient for the robot to perform the next task. There are a number of research working on CPP problem [20]–[22], however, there is no work satisfying our requirement. Therefore, we propose a variant of open CPP (VOCPP) algorithm to handle the problem.

Particularly, our main contributions are:

- An efficient algorithm, that is able to segment the steel bridge structure into steel bars and cross area regardless any kinds of input structure (tested on T-, K-, I-, L- and Cross- shape);
- An algorithm to construct an undirected graph from the input steel bridge structure data;
- VOPCP algorithm to find a shortest path for the robot to inspect all steel bars in the graph with difference between the starting and ending points;
- A method to determine whether a robot configuration belongs to free configuration, with the input as a set of cluster boundary of steel bridge structure.

III. NAVIGATION FRAMEWORK

A. Overall Architecture

As the ARA robot receives a task from the user, the task is sent to the motion planning module. The inputs for this model are point cloud data, target position and robot location from SLAM. The point cloud of the steel bridge is filtered, then projected into 2D surface, which was done in [2]. The structure segmentation algorithm separates the processed data into the clusters, then those boundaries are estimated by Non-Convex Boundary Estimation (NCBE) algorithm [2]. Graph construction algorithm processes the boundaries data to build a graph, that represents the steel bridge structure, then VOPCP algorithm solves the optimization problem to find the shortest path for robot to traverse and inspect all the available steel bars. The VOPCP algorithm can handle with any of graph with any starting and ending points. A single-query path planning such as RRT receives the cluster boundaries and shortest path as input and builds the traverse path for ARA robot to go along the steel bars. In this path planning, we propose an efficient algorithm called Point Inside Boundary Check - PIBC to determine the collision and availability of the robot configuration by the boundaries. The output of the motion planner is sent to the ARA robot controller, which regulates the robot at the lower level to perform the motion. The framework is shown in Fig. 3.

Fig. 3. The proposed navigation framework on mobile mode
B. Steel bridge Structure Segmentation

Motion planning for a robot traversing on a steel bridge is a challenging task because of the limitation of working space (steel bar’s dimensions) and the steel bridge structure’s diversity. Serious damage occurs if the robot moves out of the steel bar edges. Therefore, steel bridge perception is critical to build a motion planner. The bridge structure can be perceived by a convolutional neural network (CNN) [23], however, it requires a huge data set and expensive tasks such as labeling. Moreover, with the variety of the structures, in the best of the author’s knowledge, there is no CNN working on steel bridge structure detection.

To detect the steel bridge structure, we propose a method based on the bridge’s geometric features that they typically consist of two components: (1) steel bars and (2) cross areas. As the method segments the structure into these two components, the bridge structure is represented by a graph whose edges and vertices are the steel bars and cross areas, respectively. The steel bars are irregular in shape with one dimension being much longer than other, and their two ends connect two cross areas. The cross areas are regular in shape and its dimensions are similar, and there are at least two steel bars connected with it. The difference in the geometric features of the steel bars and cross areas are used to classify them.

From the feature analysis, an algorithm is developed based on EM-GMM classification [13], [15], [16]. The EM-GMM classification is able to separate the structure into multiple clusters with irregular shapes. However, it requires a specific cluster number as input, and if there are several types of distribution in the data, EM-GMM does not work well. To deal with unknown cluster numbers, the algorithm works on a set of cluster numbers and determines which is the most suitable number \( n_o \) for cluster segmentation. The number \( n_o \) is then inputted into the EM-GMM algorithm [13], [15] that generates a set of clusters.

Formally, to find the best cluster number for the steel bridge segmentation, a new concept - neighbor cluster - is introduced. Two clusters are considered as neighbors if they share a border with the length at least \( l_b \). From that, in Fig. 4(a), cluster 1 and 3 are considered neighbors because the border \( BC \) is longer than the threshold \( l_b \). Although there are some contacting points, there can be no neighbor-relationship between clusters 1 and 2 because the border \( AB \) is shorter than \( l_b \). The same applies to cluster 1 and 4 since \( CD \leq l_b \). From this idea, if the cluster numbers \( n_o \) are optimal, the cross area should be the one with most neighbor clusters \( n_m \) as shown in Fig. 4(b) (cluster 3). Therefore, the first idea is that the cluster number \( n_o \) is optimal if it makes the cross area cluster having \( n_m \) with highest value.

\[
0 < l_b < l_c < l_m
\]

The most neighbor cluster numbers \( n_m \) works well for \( K \), \( T \), and \( Cross \)-shape. However, for the structures such as \( L \)- and \( L \)-shapes, the highest number of neighbor is not enough to segment correctly. For \( L \)-shape, as the cluster number is more than three, there is several clusters with the same neighbor cluster number, and it is not possible to find the cross area cluster. For \( I \)-shape, there is two cross-area cluster in the structure. Thus, it is needed another feature to handle these shapes, and the difference between the most neighbor number \( n_m \) and the second most neighbor number \( n_s \) with the same \( n_c \) is considered. Combining the two features, the ratio \( r \) is defined in Eq. (1), and the highest \( r \) will be selected.

\[
r = \frac{n_m}{n_m + n_s} + \frac{n_m}{n_c}
\]  \( (1) \)

In Eq. (1), the first term tends to keep \( n_c \) small, and the second term tends to make \( n_m \) large. \( n_c \) is the number of clusters.

The procedure detail is presented by the psudo-code in Algorithm 1. The inputs are the PCL data, and a range of cluster number from \( N_{cmin} \) to \( N_{cmax} \). As \( n_c \) runs in the range of \( [N_{cmin} \) to \( N_{cmax} \)], EM-GMM algorithm segments the PCL data into a set of clusters \( S_c \). From line 5 to 7, NCBE algorithm determines the set of boundaries \( b \) corresponding to \( S_c \). The neighbor number for each cluster is determined in line 8 and from that the most and second most cluster numbers are specified. From \( n_c \), \( n_m \), and \( n_s \), the ratio \( r \) is calculated, then the \( r \) and \( n_c \) are putted into the sets \( r_n \) and \( n_{index} \), respectively. Step 13 selects the highest \( r \), which is inputted into EM-GMM algorithm to get the most appropriate segmented clusters \( S_b \).
C. Graph Construction and VOCPP

From an inspection requirement aspect, the ARA robot should monitor all available steel bars in the bridge structure to check for any failure. As the cross areas and steel bars are considered as nodes and edges, respectively, the inspection requirement is able to be solved by inspection route problem or Chinese Postman Problem (CPP) [24]. Therefore, a graph representing the structure is constructed, then the optimization problem is solved to find the shortest inspection route for the robot.

Algorithm 2 Graph Construction
1: procedure GRAPH CONSTRUCTION(data cluster set \( S_{ba} \), threshold \( d_{min} \))
2: Initialize \( E = \{ \}, V = \{ \} \), current vertex \( v_c \)
3: Calculate the center point set \( S_c \) of boundary set \( S_b \)
4: Determine neighbor matrix \( M_n \)
5: Determine borders set \( S_{bd} \) among the neighbors from \( S_c, M_n \)
6: Determine the middle point set \( S_{md} \) of \( S_{bd} \)
7: Add all vertices \( S_c, S_{md} \) into \( V = S_c \cup S_{md} \)
8: Add all edges \( E \) by connecting vertices in \( V \) if its corresponding cluster are neighbors from \( M_n \)
9: Estimate line set \( S_l \) to fit the cluster set \( S_c \) by PCA
10: Determine intersection of feature lines & their boundaries:
11: \( I_l = S_l \cap S_b \)
12: for \( i \in I_l \) do
13: for \( j \in V \) do
14: if distance from \( I_l[i] \) to \( V[j] \) larger than \( d_{min} \) then
15: Add \( I_l[i] \) into \( V \)
16: Edge \( e \) from \( I_l[i] \) to its center vertex
17: Add \( e \) into \( E \)
18: end if
19: end for
20: return \( G = (V, E) \)
21: end procedure

Algorithm 2 builds the graph from the cluster boundaries set \( S_{ba} \) (from NCBE Algorithm) and a distance threshold \( d_{min} \). Firstly, the center points \( S_c \) of clusters, neighbor matrix \( M_n \), and border middle points \( S_{md} \) are determined from step 3 to 6.

All the points in \( S_c \) and \( S_{md} \) are added into the vertices set \( V \). The edge set is built by \( M_n \) and \( V \) in step 7. After that, a set of feature’s lines that are fit to the boundaries are calculated by PCA method [25]. Step 9 determines the intersection set \( I_l \) of the feature’s lines and their cluster boundary \( S_b \). From step 11 to 19 there are dual loops to check whether the distance from a point in \( I_l \) to a vertex in \( V \) is longer than the threshold \( d_{min} \). If yes, then that point and an edge, which connects it to its center point, are added into the vertex set \( V \) and the edge set \( E \), respectively. The algorithm outputs \( G = (V, E) \), which is sent to Algorithm 3.

After getting the graph, the navigation system needs to generate the shortest path to inspect all the available steel bars. There are multiple steel bridge structure, thus the input graph could consist of Euler circuit, Euler path or none. Algorithm 3 - VOCPP is proposed to solve the optimization problem that is based on Euler Theorem [26] for Eulerian trail. The algorithm converts any input graph into an open CPP graph by adding the shortest path into the input graph and make a new one. The added path is selected by using Dijkstra’s algorithm [27]. The output is the shortest path, which visits each edge at least one, and starts and ends at the predefined positions.

The pseudo-code is shown in Algorithm 3. The inputs are the graph \( G = (V, E) \), starting and target vertex \( v_s, v_t \). Step 4 checks whether an Eulerian circuit exists in the graph. If yes, a path generated by Dijkstra’s algorithm with two vertices \( v_s, v_t \) is added into the graph. After that, the algorithm will jump to step 24. If the odd vertex set \( S_{ov} \) is not empty, step 8 checks whether \( v_s, v_t \) belong to the set. If yes, both are popped out, and a set of shortest paths is added to convert all the remained odd nodes in \( S_{ov} \) to even. Steps 11, 15 and 19 check whether the vertex \( v_s \) or \( v_t \) stays in the \( S_{ov} \), or if both are out of the set. If any input vertex is odd, it will be popped out. Vertices in \( S_{ov} \) that possess the shortest paths to the even input vertices are connected by a

Algorithm 3 Variant Open CPP
1: procedure MINIMAL PATH(\( G = (V, E), v_s, v_t \))
2: Initialize \( G_m, S_{mov}, S_{sov} \)
3: Find all odd vertices and add to \( S_{sov} \).
4: if \( S_{sov} = \emptyset \) then
5: Find shortest edges set \( E_c \) connecting \( v_s \) and \( v_t \)
6: Assign \( G_m = (V, E \cup E_c) \); go to 24
7: end if
8: if \( (v_s, v_t) \subseteq S_{sov} \) then
9: \( S_{mov} = S_{sov}\{v_s, v_t\} \), \( G_m = G \), go to 23
10: end if
11: if \( v_s \notin S_{mov} \) and \( v_t \in S_{mov} \) then
12: select a vertex \( v_s \in S_{ov} \) which creates the shortest path \( E_c \) connect \( v_s \) to it
13: \( S_{mov} = S_{sov}\{v_s, v_t\} \), \( G_m = (V, E \cup E_c) \), go to 23
14: end if
15: if \( v_s \in S_{mov} \) and \( v_t \notin S_{sov} \) then
16: select a vertex \( v_t \in S_{ov} \), which creates the shortest path \( E_c \) connect \( v_t \) to \( v_s \)
17: \( S_{mov} = S_{sov}\{v_s, v_t\} \), \( G_m = (V, E \cup E_c) \), go to 23
18: end if
19: if \( v_s \notin S_{mov} \) and \( v_t \notin S_{sov} \) then
20: select two vertices \( v_{s1}, v_{t2} \in S_{ov} \) which creates two paths \( E_{d1}, E_{d2} \) that sum of them is shortest
21: \( S_{mov} = S_{sov}\{v_{s1}, v_{t2}\} \), \( G_m = (V, E \cup E_{d1} \cup E_{d2}) \), go to 23
22: end if
23: Using \( G_m \), find a set of edges \( E_a \) whose sum are shortest to convert all vertices in \( S_{mov} \) to even vertices.
24: Find the Eulerian path \( P_{robot} \) from \( G_m = (V, E_m \cup E_a) \)
25: return \( P_{robot} \)
26: end procedure
Algorithm 4 Point Inside Boundary Check

1: procedure SAMPLE CHECK(Sbo, c, P)
2: Calculate the cluster center point c_p of each boundary in Sbo
3: Project from configuration c to the robot position set PS
4: Initialize boolean set bs[|len(PS)|] = Fail
5: for i ∈ (0, len(PS)) do
6: Calculate distances d_s from PS[i] to c_p
7: Select n center points, which are closest to PS[i]
8: for p ∈ (0, n) do
9: Find m_p closest points to PS[i] in each Sbo[p]
10: for j ∈ m_p do
11: Calculate distance d_m_p from m_p[j] to c_p
12: if d_s < d_m_p then
13: bs[i] = True
14: end if
15: end for
16: end for
17: end for
18: for i ∈ (0, len(PS)) do
19: if bs[i] = Fail then return Fail
20: end if
21: end for
22: return True
23: end procedure

shortest path to convert the even input vertex into odd. After that, the selected vertex is also popped out of Sbo. All then go to step 23 to convert all the odd vertices in the remaining Sbo into even vertices by Dijkstra’s algorithm. Step 23 will find the shortest path that traverses all edges of the graph by Fleury’s algorithm [26].

D. Point Inside Boundary Check - PIBC

As receiving the shortest path from the VOCP algorithm, a single-query motion planning is deployed to generate path to control the robot traverse all the defined edges. It will plan the motion step by step one edge each time.

In our scenario, the robot moves on a set of steel bars with assumption that there is no obstacle on it. The free configuration, however, is limited by the space of the steel bar surface. The serious damage can occur if the robot moves out of the steel bar edges. It is a hard constraint to construct a free configuration C_free for the robot. To utilize all available configurations, the point cloud boundary is applied to build the obstacle-free configuration. After receiving the boundary set Sbo from NCBE algorithm, a robot configuration c_i belongs to the free configuration C_free if all of its projected points Spp lie inside one of the steel bar boundaries b_i ∈ Sbo.

To determine whether a point lies inside a steel bar boundary, a geometric technique named center-closest points is applied as shown in Fig. 5. A sample point P is considered inside the boundary if its distance d_s to the center point C is shorter than the distances d_m of its closest neighbors to the center. The closest neighbor set N is determined by finding the minimal Euclidean distance set.

The implementation of the technique is presented in Algorithm 2. The inputs are the boundary set Sbo, robot configurations c, and robot parameter set P. From step 5 to 7, the loop runs through all the robot configurations. For each configuration, it finds a set of clusters in which it possible belongs to. For each cluster, the technique center-closest points is deployed to specify whether it lies inside that boundary. If the configuration stays inside only one of the cluster boundary, its corresponding boolean variable bs[] will set True. Step 18 uses a for loop to check whether there is any robot configuration not belonging to any cluster boundary. The algorithm returns True if any configuration lies inside a boundary. To reduce the processing time for the robot, RRT motion planning [28], [29] is deployed in the robot.

IV. EXPERIMENT RESULTS

We implemented the experiments on the ARA robot, which ran on the setup of steel bridge structures such L-, Cross-, T-, K, and I- shape. Due to the lab condition, we combined several steel bars on the ground to make the shape types mentioned as shown in the first row of Fig. 6. To present the experiment result succinctly, the structure images and point cloud data are rotated 90 degree with the viewpoint from the right to left. All the algorithms integrated into ARA-robot control program and ran on ROS.

A. Structure Segmentation and Boundary Estimation

The result of the segmentation algorithm is shown in Fig. 6. The images in the first row are the point cloud data of Cross-, K-, L-, T-, and I- shapes after filtering and projected into the robot coordinate frames. The view-point of camera is from right to left, and the farther to the camera, the more degrading of sensor data quality. The efficiency of Algorithm 1 is shown in the third row of Fig. 6. The algorithm runs well and is able to segment properly the cross areas and the steel bar parts, except the I- shape. The reason is that there are two cross areas in I-shape structure, which make algorithm 1 confuse. This problem will be improved in the future research. By a robust graph construction algorithm, however, the graph of I- shape can be still built and help generate the path for the robot. After segmenting, the clusters are sent to the NCBE algorithm [2], which works efficiently to give back the boundaries for the cluster.

B. Graph Construction and VOCP Algorithm

Graph Construction (Algorithm 2) and VOCP (Algorithm 3) process the boundary data from the NCBE algorithm, and output the result on the fourth and fifth rows, respectively. In the fourth row, the graph is built based on the center point, edge points of the boundary and the border points between the neighbor clusters. The graphs cover the steel bars’ length for most structures, except the Cross- structure (Fig. 6p). It is because of the distance from the center points to the corresponding edge points are too close, the depth sensor can cover all the distance. Therefore, it is no need an edge to connect that two points. In Fig. 6q, the edge (12-3) does not go along with the steel bar but crossing on one edge. It occurred because the data quality is not good, and influencing the line fitting algorithm. The same reason happens to the T- structure in Fig. 6r.

The figures in the last row of Fig. 6 show the shortest path generated by the VOCP algorithm. Due to the probabilistic
Fig. 6. The images of input structures (a-e), the corresponding point clouds (f-j), the segmentation (k-o), boundary estimation and graph construction (p-t), and the shortest path (u-y).

Fig. 7. The experiment on ARA robot on L-shape structure.
feature of EM-GMM algorithm, the cluster indices change each time running. Starting at a random vertex $v_s$, the robot follows the arrow lines to the next vertex, and comes back if there is a dead end. The route ends at the predefined ending vertex $v_t$, and the generated paths are optimal with shortest length. Again, due to the point cloud data quality, in Fig. 6,x, the edges (3,12), (2,11) are not possible for robot to traverse. To prevent the robot go on the edges, the motion planner needs to handle this case.

C. Point Inside Boundary Check - PIBC and Path Planning

Using PIBC algorithm with RRT motion planner, a motion path is generated as shown in Fig. 7.h. The algorithm is significantly affected by the data quality, thus the data should be collected frequently. All the above algorithms are applied to the ARA robot, which is tested on the L-shape structure as shown in Fig. 7.b,c,d. The width of the steel bars is almost double comparing to the robot width. The robot can make a sharp turn at the cross area. As experimenting on complex structures, there is a technical difficulty on localization of close area, which is bind to the depth sensor and the control of the robot to make a sharp turn.

V. Conclusion and Future Work

A navigation framework is proposed for the ARA robot to run on mobile mode. In this mode, the robot needs to cross and inspect all the steel bars. The most significant contributions of this research are four algorithms, which can process the depth data, then output a traverse path for the robot. Algorithm 1 - Structure Segmentation segments the steel bar structures into two sets: steel bars and cross areas. Based on the segmentation result, the graph construction - algorithm 2 builds a graph that represents the bridge structure. From the graph, algorithm 3 - VOCPG generates a shortest path for the robot to move and inspect all the available steel bars. Algorithm 4 helps RRT path planning generating a trajectory, which the robot controller regulates the robot to follow.

There are several aspects, which needs to be improved in our research. The efficiency of algorithms needs to extend to process more bridge structures. The stability of the segmentation algorithm, which depends on the probability method, is not in well-operating and outputs sometime inappropriate segmentation. The drawbacks will be solved in the next research, which can help the robot operate in fully autonomous navigation.

References

[1] Son T Nguyen, Anh Q Pham, Cadence Motley, and Hung M La. A practical climbing robot for steel bridge inspection. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 9322–9328. IEEE, 2020.
[2] Hoang-Dung Bui, Son Nguyen, UH Billah, Chunle Ng, Alireza Tavakkoli, and Hung M La. Control framework for a hybrid-steel bridge inspection robot. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, Nevada, USA, October 25 – 29, 2020.
[3] TaeWon Seo and Metin Sitti. Tank-like module-based climbing robot using passive compliant joints. IEEE/ASME Transactions on Mechatronics, 18(1):397–408, 2012.
[4] Hongqiang Wang and Akio Yamamoto. Analyses and solutions for the buckling of thin and flexible electrostatic inchworm climbing robots. IEEE Transactions on Robotics, 33(4):859–900, 2017.
[5] S. T. Nguyen and H. M. La. Roller chain-like robot for steel bridge inspection. 2019.
[6] S. T. Nguyen and H. M. La. Development of a steel bridge climbing robot. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1912–1917, 2019.
[7] Keith D Kotay and Daniela L Rus. Navigating 3d steel web structures with an inchworm robot. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96, volume 1, pages 368–375. IEEE, 1996.
[8] Yang Xie, Wei Wang, Jiajie Guo, and Kok-Meng Lee. Edge detection using structured laser pattern and vision for mobile robot navigation. In 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pages 910–915. IEEE, 2011.
[9] David Pagano and Dikai Liu. An approach for real-time motion planning of an inchworm robot in complex steel bridge environments. Robotica, 35(6):1280–1309, 2017.
[10] R. Deits and R. Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In Algorithmic foundations of robotics XI, pages 109–124. Springer, 2015.
[11] S. Jatsun, S. Savin, and A. Yatsun. Footstep planner algorithm for a lower limb exoskeleton climbing stairs. In International Conference on Interactive Collaborative Robotics, pages 75–82. Springer, 2017.
[12] A-C. Hildebrandt, R. Wittmann, F. Sygulla, D. Wahrmann, D. Rixen, and T. Buschmann. Versatile and robust bipedal walking in unknown environments: real-time collision avoidance and disturbance rejection. Autonomous Robots, 43(8):1957–1976, 2019.
[13] Douglas A. Reynolds. Gaussian mixture models. Encyclopedia of biometrics, 741, 2009.
[14] Tim Pfeifer and Peter Protzel. Expectation-maximization for adaptive mixture models in graph optimization. In 2019 International Conference on Robotics and Automation (ICRA), pages 3151–3157. IEEE, 2019.
[15] Thanh Minh Nguyen and QM Jonathan Wu. Fast and robust spatially constrained gaussian mixture model for image segmentation. IEEE transactions on circuits and systems for video technology, 25(4):621–635, 2012.
[16] Konstantinos Blekas, Aristidis Likas, Nicolas P Galatsanos, and Isaac E Lagaris. A spatially constrained mixture model for image segmentation. IEEE transactions on Neural Networks, 16(2):494–498, 2005.
[17] Hunter Goforth, Xiaoyan Hu, Michael Happend, and Simon Lacey. Joint pose and shape estimation of vehicles from lidar data. arXiv preprint arXiv:2009.03964, 2020.
[18] Gabriela Zarzar Gandler, Carl Henrik Ek, Mårten Björkman, Rustam Stolkin, and Yasemin Bekiroglu. Object shape estimation and modeling, based on sparse gaussian process implicit surfaces, combining visual data and tactile exploration. Robotics and Autonomous Systems, 126:103433, 2020.
[19] Stefan Kraemer, Mohamed Essayed Bouzouara, and Christoph Stillier. Simultaneous tracking and shape estimation using a multi-layer laser scanner. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pages 1–7. IEEE, 2017.
[20] Harold Thimbleby. The directed chinese postman problem. Software: Practice and Experience, 33(11):1081–1096, 2003.
[21] Horst A Eiselt, Michel Gendreau, and Gilbert Laporte. Arc routing problems, part i: The chinese postman problem. Operations Research, 43(2):231–242, 1995.
[22] Dino Ahr and Gerhard Reinelt. A tabu search algorithm for the min–max k-chinese postman problem. Computers & operations research, 33(12):3403–3422, 2006.
[23] Yasutaka Narazaki, Vedhus Hoskere, Tu A Hoang, and Billie F Spencer Jr. Automated vision-based bridge component extraction using multiscale convolutional neural networks. arXiv preprint arXiv:1805.06042, 2018.
[24] Jack Edmonds and Ellis L Johnson. Matching, euler tours and the chinese postman. Problems, part i: The chinese postman problem. 5(1):88–124, 1973.
[25] Konstantinos Blekas, Aristidis Likas, Nikolas P Galatsanos, and Isaac E Lagaris. A spatially constrained mixture model for image segmentation. IEEE transactions on Neural Networks, 16(2):494–498, 2005.
[26] Stefan Kraemer, Mohamed Essayed Bouzouara, and Christoph Stillier. Simultaneous tracking and shape estimation using a multi-layer laser scanner. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pages 1–7. IEEE, 2017.
[27] Harold Thimbleby. The directed chinese postman problem. Software: Practice and Experience, 33(11):1081–1096, 2003.
[28] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7):846–894, 2011.

[29] Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms. arXiv preprint arXiv:1808.10703, 2018.