Autonomous mobile robot path planning to inspect and repair heavy steel plates

Sara Roos-Hoefgeest
ISA
Universidad de Oviedo
Gijón, Spain
roossara@uniovi.es

Álvaro Fernández García
ISA
Universidad de Oviedo
Gijón, Spain
fernandezgalvaro@uniovi.es

Ignacio Alvarez Garcia
ISA
Universidad de Oviedo
Gijón, Spain
ialvarez@uniovi.es

Rafael C. Gonzalez
ISA
Universidad de Oviedo
Gijón, Spain
rcgonzalez@uniovi.es

Abstract— **Currently, the inspection and repair of heavy steel plates are performed by human workers. Repair tasks usually involve long hours in awkward positions that can cause physical problems to the worker. This paper proposes a mobile robot to inspect surface defects in a big sized steel sheet, previously delimited by a white contour, and then plan the trajectories to repair them. For safety reasons, the robot cannot leave the sheet, although it must cover the entire surface. We propose path planning algorithm to inspect the sheet, whose orientation and dimensions are unknown a priori, and to repair the defects at any location on the sheet, with a grinding tool attached to the mobile robot. For this purpose, each defect is assigned an orientation associated with the robot's direction of movement that ensures safety. We validate the proposal through experiments using an omnidirectional robot.**

Keywords— Automated Coverage Path Planning, AMR Inspection and Grinding, Vision Guided Robotics, Mobile Robot, ROS

I. INTRODUCTION

In an increasingly automated industry, mobile robots appear in a growing range of applications. One example is the deployment of autonomous mobile robots that apply Non Destructive Evaluation (NDE) techniques to detect surface defects. Some examples may be found at [1], [2] and [3]. Other applications try to replace human workers when the task involve important occupational hazards, such as grinding. Those applications are traditionally solved with robotic manipulators as in [4].

In the production of heavy steel plates, both problems appear concurrently. A human inspector analyses the plate and encircles all the defects found. Then, another worker manually removes all defects using an angle grinder. The worker usually needs to adopt awkward postures, increasing the range of occupational hazards to which he is exposed.

Our work aims to reduce those risks by automating this task. A vision guided omnidirectional mobile robot scans the steel sheet to look for the encircled defects marked by a human expert. As a result, we get a map of the plate that includes all the defects. The robot uses this map to plan a path to repair the defects using a grinding tool placed at a fixed position. The robot is required to operate within the plate for safety reasons.

In this paper, we describe methods to map the plate and to ensure that all defects are completely covered by the grinding tool while the robot moves within the plate. Defects may appear anywhere, including the edges of the plate, and may have very different areas. The thickness of heavy steel plates ranges from 5 to 150 mm. Their width varies from 1400 to 3300 mm and their length may change from 4 to 18 m.

In industrial inspection tasks, it is usually required that the full area is covered. In a robotic application, this leads to the problem known as Coverage Path Planning (CPP). The paper by Galceran and Carreras [5] provide a comprehensive review of the problem.

One family of CPP algorithms discretize the workspace into a grid. Typically, each element of the grid is a square the same size of the robot or the tool. The Spiral-STC algorithm [6] or Yang and Luo neural network based approach [7] are examples of this group of algorithms.

The other family of algorithms decomposes the free space into smaller regions that are easier to cover. Examples of this type of methods are the trapezoidal [8], boustrophedon [9], Morse [10] or slice decompositions [11].

La et al. [1] proposed a boustrophedon based strategy to inspect a bridge deck using a mobile robot. Three GPS waypoints define the rectangle to be covered. Sensors are in contact with the robot, which can safely move outside of the working space to complete the scan. Gui and Li [2] proposed a similar approach to analyze airport runways using an omnidirectional robot. Usually, the result of the inspection is a mosaic image resulting from the composition of all the partial information gathered while travelling the working area.

Lim et al. [3] proposed a grid based algorithm to inspect bridge decks. The robot is equipped with a Pan-Tilt-Zoom camera so that the viewed area is outside the robot footprint. They considered three possible camera orientations with respect to the robot and four possible robot orientations. They proposed a genetic algorithm to find a route that allow covering the working area with the camera while minimizing the number of turns and the traveling distance.

None of the previous methods imposes that the robot must move inside the working area. This requirement leads to the "milling problem", a particular version of the CPP problem where the cutting tool is not allowed to exit the working area. Macleod et al. [12] proposed the use of CAD/CAM software for path planning and then use a post-processor to adapt the path and control the robot.

Deng et al. [4] proposed a vision based system to repair surface defects on large-scale buoyancy modules. In this work, an inspector marks with an ellipse the defects. Using computer vision, a robot arm searches the ellipses and plans the trajectory to repair them. The robot follows a spiral path starting at the center and moving outwards.

The paper is organized as follows. Section II describe the algorithms used to compute the inspection path and a way to decompose the scanned sheet in four different working zones to guarantee that the robot remains inside the sheet while is in operation. Section III shows results achieved with our prototype at the lab. Finally, section IV presents our conclusions.

Research supported by: Daorje S.L.U inside project code CN-18-011 financed by Asturias Regional Government.

II. METHODS

A. Hardware

Fig. 1 shows a front and a rear view of the robot used to solve this problem. The base is an omnidirectional Summit XL Steel robot from Robotnik. It mounts an AAK angular kit from FerRobotics as a grinding tool. It is mounted out of the footprint of the base, positioned at the front of the robot on the left side. Two fixed Intel RealSense D435 RGB-D cameras provide the information to reconstruct the plate surface. Both are oriented 30º to the ground and placed, one on the front and one on the back of the robot.

(a) (b)

Fig. 1. Mobile Robot used in experiments: (a) Front view and (b) rear view.

The robot is equipped with additional laser sensors for localization and obstacle detection. Localization is performed with a SLAM algorithm using the *gmapping* ROS package [13]. To detect possible obstacles, a 3D map of the environment is generated using the *octomap_mapping* ROS package [14].

B. Process Workflow

As in [4], our problem starts after an inspector has analyzed a heavy steel plate and marked every defect using a high contrast marker, for example white chalk. The curves that encircle the defects do not have any specific shape. In addition, they may be slightly opened or touching the edge.

Fig. 2 shows a schematic view of the overall process. First, the robot has to ensure a complete inspection of the plate without moving out of it. The plate has a rectangular shape, but its dimensions and the robot pose with respect to the plate are unknown in advance. Then, the robot maps all marked defects and plans a grinding path for each one. Planned paths must cover all the defective area while the robot keeps moving on top of the plate. When a human operator selects one of the defects, the robot navigates to it and execute the planned path. We assume that the plate is free of obstacles, so the path planning algorithm do not need to take them into account. If navigation sensors detect an obstacle within the plate, the robot will stop until the obstacle disappears. This is to ensure a safe operation within an industrial environment.

C. Inspection process

1) Sheet plane calibration

To identify the sheet plane, an ArUco marker [15] [16], is placed on the sheet. Its corners are the reference points to solve for the homography that relates the sheet and the image. It is necessary to ensure that only the plate is visible in the image area closest to the camera, since these points will be the ones considered to calculate the plane. This calibration process is carried out for both cameras.

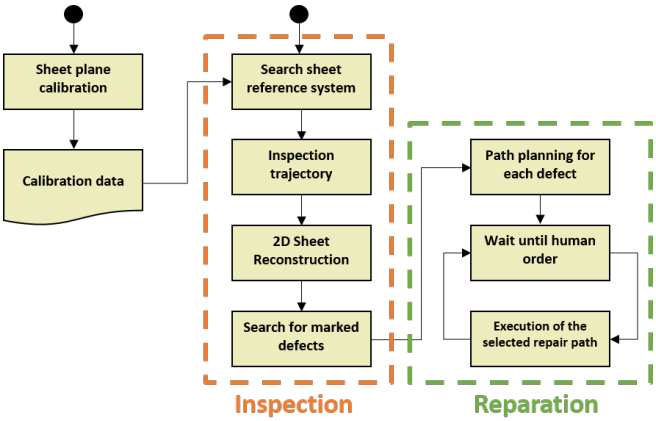


Fig. 2. Process workflow. In orange, the inspection process. In green, the reparation path planning process.

As the size of the ArUco marker is known, we can compute the homography to solve for the perspective transformation that relates both planes (Fig. 3). This homography is also used to construct the zenith view of the complete sheet. In this process, we also obtain the pixel-meter ratio for the transformed camera images.

The RGB-D camera provide the 3D position of every point in the image. For a robust computation of the plane parameters, we take 1 out of every 10 points from the lower 2/3 portion of the image and assume that they belong to the plane. The condition that a point ($[X_i \quad Y_i \quad Z_i]^T$) belongs to a plane π ($[a \quad b \quad c \quad d]^T$) can be easily stated using projective geometry, because their inner product should be zero. Applying this condition to all the points, we get a system of equations of the form:

$$\begin{bmatrix} \vdots & & & \\ X_i & Y_i & Z_i & 1 \\ \vdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0 \qquad (1)$$

This system is easily solved applying SVD decomposition. The distance of a point to the plane is computed as:

$$\Delta = \frac{a\,X + b\,Y + c\,Z + d}{\sqrt{a^2 + b^2 + c^2}} \qquad (2)$$

To know which points belong to the sheet in a new RGB-D image, we compute their distance to the plane using (2). If the distance is lower than a predefined threshold, the point belongs to the sheet. The threshold is proportional to the standard deviation of the values Δ_i computed for the points used in (1). Only the closest object to the camera is considered as part of the plate since there may be points of other objects that do not belong to the sheet but lie on the same plane. Fig. 4 shows some results.

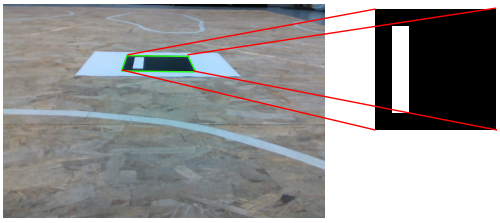


Fig. 3. Compute homography that transforms the camera plane to the sheet plane with an ArUco.

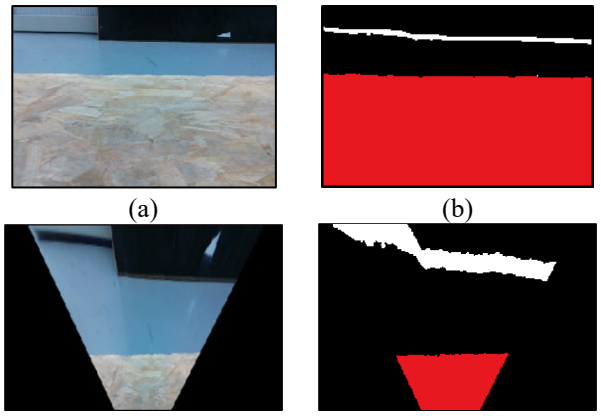


Fig. 4. Sheet plane calibration results. (a) RGB image (b) Points of image (a) that belongs to the sheet plane (in red). (c) Zenith view of (a) computed using the estimated homography. (d) Points of image (c) that belongs to the sheet plane (in red).

2) Sheet reference system definition

The robot do not have a map of the plate, which will become its working area. The only previous knowledge is that the robot is on top of the plate. In order to build a map of the plate, it is necessary to define a coordinate frame associated to the plate and referenced to the world coordinate frame. We chose to place the origin in one of the corners of the sheet.

The robot uses the information provided by the RGB-D cameras to locate the corner. Images are transformed to zenithal view and only the region corresponding to the plate is processed. Algorithm 1 outline the procedure to define the coordinate plate associated to the plate.

Algorithm 1: Define plate reference frame

Step 1. Move backward until a plate edge is located.
Step 2. Rotate to align the edge with the horizontal scanline of the camera.
Step 3. Move to the right until a vertical edge of the plate is located.
Step 4. The origin of the coordinate frame is the intersection of both edges. The X axis is aligned with the vertical edge and the Y axis is aligned with the horizonal edge.

3) Inspection trajectory path

Once the coordinate frame is defined, the robot moves to the starting point, which is located at a predefined distance of the origin. In this way, we guarantee that the robot is on top the plate and the edges that define the limits of the plate are within the field of view.

We follow a boustrophedon type path to inspect the whole plate as shown in Fig. 5. The inspection path starts by following the side edge of the sheet until the front edge is found. Then, the robot moves laterally following the front edge a previously computed distance, according to camera FOV, to ensure the lateral overlapping of the images of the different scans. If the robot detects a new lateral edge before the lateral movement is complete, it will stop when a predefined distance to that edge is achieved. The robot follows the new edge until the surface is completely scanned.

As the robot moves, both RGB-D cameras are used to capture surface images and to control robot movement. Edges are detected and tracked to constrain robot movement. When the robot moves through the middle of the sheet, it may have no reference to the edges. In this case, the path is guided by the robot's localization system, ensuring that the robot's orientation is parallel to the edge of the sheet, whose reference system had been previously set.

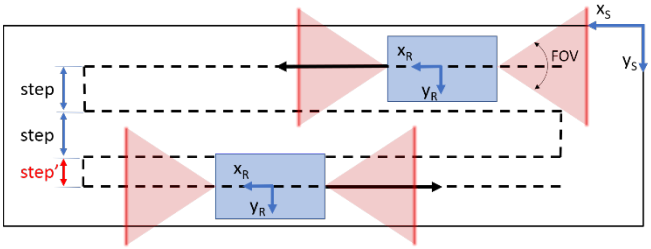


Fig. 5. Inspection trajectory of the sheet. The robot (in blue) moves along the sheet according to the information provided by the RGB-D cameras. Red triangles represent the field of view of the cameras.

4) 2D Sheet reconstruction and search for marked defects.

Once the robot has travelled the entire sheet, we have a set of images for each camera. These are transformed to remove perspective distortion and a mask to segment plate / not plate points is computed. To get an image of the overall sheet, we build a panorama using both sets of images. A Template Matching approach suffices to compute the translation between consecutive images.

Once the sheet is reconstructed, we look for possible marked defects. The only assumption is that defects are encircled using a high contrast marker, for example chalk. Detected defects are mapped as polygons with respect to the reference system of the sheet.

As a human inspector marks the curve delimiting the defect, this may lead to slightly opened curves. To solve this problem, a morphological closing operation is applied to close the detected white forms that are slightly open. We look for contours in the resulting closed image. In the case of contours found at the edges of the sheet, we consider that the edge itself is a limit for the defective area, as shown in Fig. 6. Algorithm 2 describes the procedure.

Algorithm 2: Search for marked defects

Step 1. Threshold image to search for white shapes.
Step 2. Apply morphological closing operation to close the detected white forms that are slightly open.
Step 3. Find contours. Split into closed and open contours.
Step 4. Close open contours that are near to the sheet edges considering the edges as part of the contour.
Step 5. Close all other open contours by connecting their ends.
Step 6. Approximate all detected contours by a polygon using the Ramer-Douglas-Peucker (RDP) algorithm [17]. This algorithm aims to reduce the number of points used in the approximation of a curve.
Step 7. Convert the coordinates of its vertices from pixels to meters.

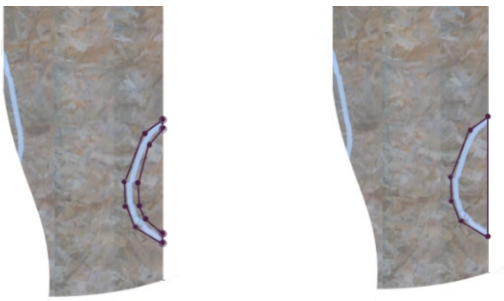


Fig. 6. Closing open contours on sheet edges.

D. Grinding path planning

The geometry of the areas to be repaired is used for path planning. Two requirements are key to solve this problem. First, the robot cannot leave the metal sheet for safety reasons. Second, defects may touch the edges of the metal sheet. Looking at the example in Fig. 6 it is obvious that if the tool is within the footprint defined by the robot wheels, it is impossible to cover the defect while the robot remains within the plate. Assuming that the tool is in a fixed position outside the robot footprint, it is the robot orientation what will define whether it is possible to cover the entire defect or not.

1) Division of the sheet into work zones

To solve this problem, we compute which zone of the plate the tool can reach while the robot operates within the plate with a constant orientation. One way to compute the solution is to use morphological operators. To make this computation, we first compute the area the robot may travel while the robot and the tool remain within the plate. Let $\mathcal{P}$ be the set of points that belong to the plate referred to its center. Let $\mathcal{R}$ the set of points that define the robot footprint and $\mathcal{T}$ the center of the tool, both referred to the center of the robot (see Fig. 7). The solution, $\mathcal{S}_R$, is the erosion of the plate ($\mathcal{P}$) using the union of the robot and the tool ($\mathcal{R} \cup \mathcal{T}$) rotated to achieve the desired orientation (θ):

$$\mathcal{S}_\mathrm{R} = \mathcal{P} \ominus \left(rot(\mathcal{R} \cup \mathcal{T}, \theta)\right) \tag{3}$$

To compute the area reached by the tool while the robot is completely within the plate we dilate $\mathcal{S}_R$ using the tool with the desired orientation:

$$\mathcal{S}_\mathrm{T} = \mathcal{P} \oplus \left(rot(\mathcal{T}, \theta)\right) \tag{4}$$

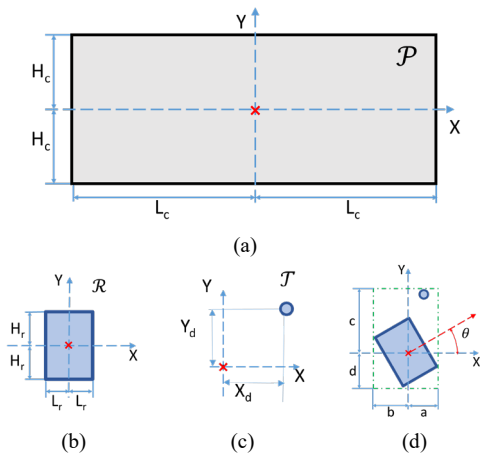


Fig. 7. (a) The heavy steel plate as a set. The reference point to describe the set is the center of the plate. (b) Robot footprint is also a rectangular set. The reference point is at the center. (c) The tool is decribed by its center and is referenced to the robot center. (d) The union of the robot footprint and the tool. The robot is rotated θ degrees with respect to X axis and the reference is the robot center. Its bounding box is a rectangle aligned with the plate with a length (a+b) and a width (c+d) and a, b, c, d are positive values.

The result is shown in Fig. 8. $\mathcal{S}_R$ is marked in orange. It is a rectangle bounded by equations $x = L_c - a$, $x = b - L_c$, $y = H_c - c$ and $y = d - H_c$. If we consider the tool as a single point, $\mathcal{S}_\mathrm{T}$ is equal to $\mathcal{S}_R$ but translated by $\boldsymbol{t} = R_\theta \begin{bmatrix} X_d \\ Y_d \end{bmatrix}$, where R_θ is the rotation matrix associated to the change in orientation of the robot. Whenever the components of the translation $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$ are smaller than a, b, c or d, then $\mathcal{S}_T$ will not touch the corresponding plate edge. In the example shown in Fig. 8, the rotated tool lies on the first quadrant and $t_x \leq a$. Therefore, the right side of $\mathcal{S}_\mathrm{T}$ do not touch the right edge of the plate because the translation is not enough to compensate the shrinking of $\mathcal{S}_\mathrm{R}$ with respect to the plate.

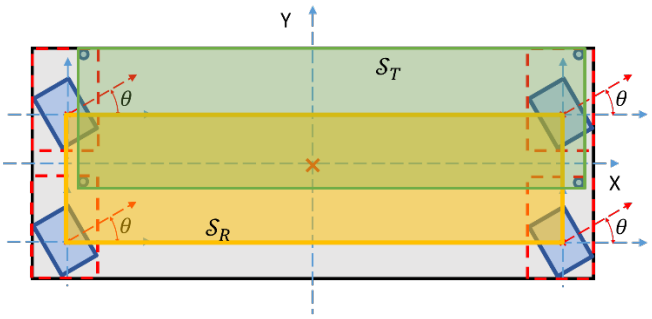


Fig. 8. The golden rectangle is the area travelled by the center of the robot while the robot and the tool remain within the plate ($\mathcal{S}_R$), when the robot is rotated θ degrees. The green rectangle is the area covered by the tool ($\mathcal{S}_\mathrm{T}$) for the same orientation.

As the plate has a rectangular shape, we consider four possible orientations aligned with the plate axis: 0º, 90º, 180º and 270º. The combination of the four orientations allows covering the whole plate. The solution for orientation $(180° + \theta)$ is equal to the solution for θ reflected with respect to the center of the plate, as shown in Fig. 9. A more detailed description of the calculations and the results can be found in [18].

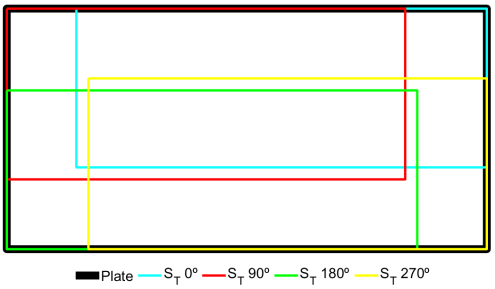


Fig. 9. $\mathcal{S}_T$ for orientations 0º (cyan), 90º (red), 180º (green) and 270º (yellow). The border of the plate is marked in black. The union of the four orientations covers the full plate surface.

2) Defect decomposition and path planning

Algorithm 3 describes the process to generate the tool trajectories that allow repairing a defect. We assign to each defect a robot orientation according to the working zone where the defect is included. If a defect spans over several working zones, we split it so that each part is within a single working zone (see Fig. 10).

Algorithm 3: Defect decomposition into simple forms

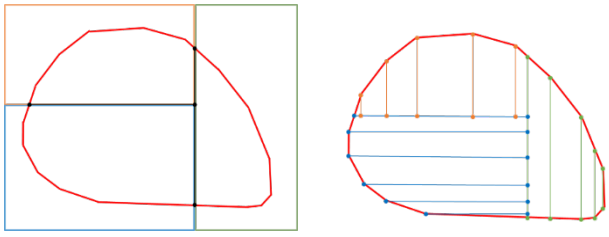
Step 1. Divide the original defect polygon in smaller polygons ensuring that each polygon belongs to a single working zone.

Step 2. Trapezoidal decomposition of the subpolygons resulting from step 1, according to to the direction of movement of the robot based on the work zones.

- **Step 2.1.** Check the intersections that a line in the direction of movement through each of the vertices. Add it as a new vertex.
- **Step 2.2.** Building the trapezoids of the decomposition using the original vertices and the vertices obtained in step 2.1

Step 3. Generate a boustophedron type trajectory to cover each trapezoid. The distance between lines parallel to the advance movement is equal to the efective size of the tool.

Step 4. Translate the tool trajectory to get the robot trajectory.



(a) (b)

Fig. 10. Initial steps of the path planning proccess. In the example, a defect (in red) spans over three working zones (blue, orange and green rectangles). (a) After Step 1, the defect is splited in 3 new polygons resulting of the intersection of the defect with each working zone. (b) In Step 2 a trapezoidal decomposition is applied to each subpolygon. The direction of the decomposition is defined by the orientation associated to each working zone.

To generate a boustrophedon type trajectory, we apply a trapezoidal decomposition [19] to the polygon that describe the defect. The polygon has an associated robot orientation. For each vertex, we consider the line parallel to the robot orientation and contains the vertex. This line will intersect the polygon in a point that becomes a new vertex. The vertices associated to two neighboring lines define each trapezoid. As an example, consider the polygon resulting from the intersection of the initial curve with the working area marked in blue in Fig. 10 (a). Assume that robot orientation for this working area is parallel to the horizontal axis. We trace a horizontal line through each original vertex. Its intersection with the polygon is a new polygon vertex (blue dots in Fig. 10 (b)).

Tool trajectory is formed by paths parallel to robot orientation and spaced a distance lower than the tool width. To obtain the robot path we only need to apply to the tool path the translation that exists between the tool and the robot reference system (Fig. 11).

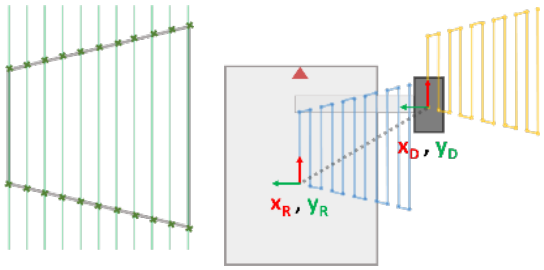


Fig. 11. On the left, parallel trajectories that guarantee the total coverage of a trapezoid. On the right, the tool (dark gray) follows the orange path to cover the trapezoid shown on the left. In blue, the translated path to be executed by the robot (light gray).

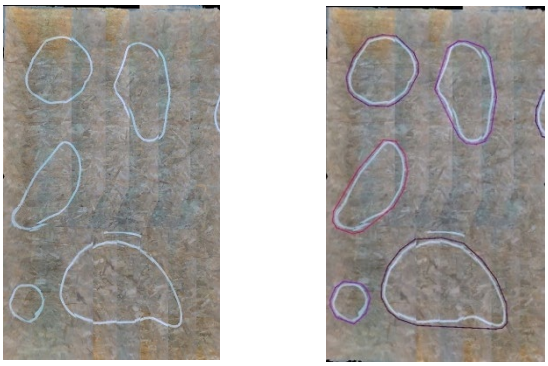
Local obstacle avoidance is not considered in the path generation, as the sheet must be completely clear for safety reasons. If an obstacle is detected inside the plate, the robot will stop its operation immediately.

III. Experiments and Results

The robot described in section II.A has been used to test our algorithms. A sheet of wood was used to test the algorithms. The sheet was 3.7m long, 2.5m wide and 25mm thick. Defective areas were marked using white tape. The software has been developed for ROS-Kinetic [20] running on a Ubuntu 16.04 LTS operating system. We used the open source OpenCV library [21] to process the images. Algorithms are coordinated using the FlexBE software [22].

A. Inspection

Fig. 12 (a) shows a panoramic reconstruction of the 2D sheet from the images taken by front and rear RGB-D cameras while scanning the sheet. The resulting panorama is processed to find the curves that delimit the defects. Each detected curve is approximated by a polygon (see Fig. 12 (b)). The results are referenced to the plate reference frame, located in a corner of the sheet. This reference point is searched before the scanning process starts. Fig. 13 (a) shows the visualization of the world map in Rviz that includes the robot, the sheet and the defects identified. The environment has been mapped using *gmapping* and *octomap_mapping* ROS packages. The sheet and the polygons that encircle the defects have been mapped using our scanning algorithm.

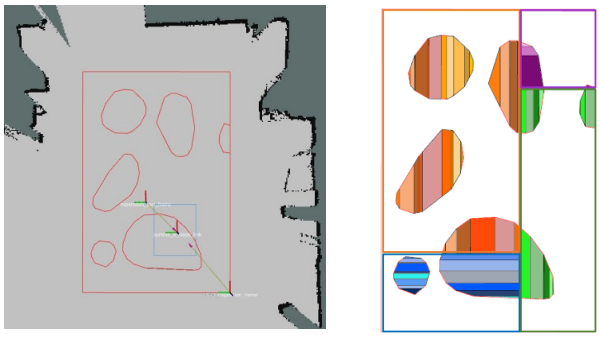


(a) (b)

Fig. 12. (a) 2D sheet reconstruction using Template Matching. (b) Polygons that approximate the defects marked by the human inspector.

B. Grinding path planning

Once the information of the images is extracted, the decomposition can be performed. Fig. 13 (b) shows how the sheet is successfully divided in 4 working zones and the trapezoidal decomposition according to the orientation for each working zone. With the zones for reparation correctly defined, the state machine implemented using FlexBE can now calculate the paths to repair the flaws in the sheet. This process is carried out ensuring a complete coverage of the defect and considering the different orientations that guarantee not to leave the surface. Fig. 14 (a) shows the tool trajectories for the largest defect. It spans three different working zones and therefore the trajectory is divided in three different parts. This effect clearly visible in Fig. 14 (b), where the trajectories of the robot center for the same defect are shown in blue.



(a) (b)

Fig. 13. Images obtained from Rviz. (a) Visualization of the environment, with the robot on the sheet metal and the defects found. (b) Working zones for the reparation and the trapeziums results of de decomposition. The decomposition according to work areas is represented with different color palettes.

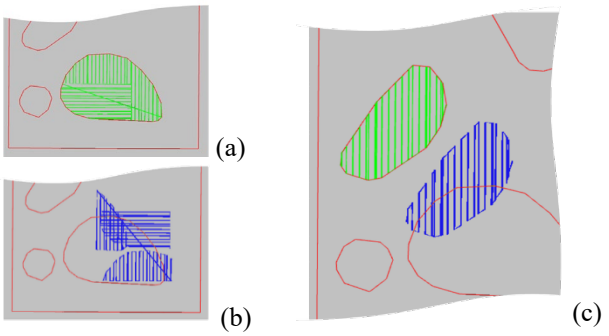


Fig. 14. Images obtained from Rviz to show the reparation paths for two different defects. Tool trajectory is represented using the green color in (a) and (c). The robot path is planned so that the path followed by the fixed tool fits perfectly to the defect area. In (b),we can see that the robot follows three different paths because it has to change its orientation according to the different working zones. In (c), a single orientation suffice and a single trajectory is generated.

IV. Conclusion

We have proposed a system for inspecting a sheet of unknown dimensions and orientation, with a robot located anywhere on it. The robot detects marked defects on any place of the sheet and generates safe trajectories to repair them. For safety reasons, the robot cannot leave the sheet during the inspection or repair phases. The experiments show very satisfactory results that validate the proposed algorithms.

It is essential to detect the sheet metal, as the robot cannot leave it. The proposed sheet plane identification provides very good and robust results. Other existing algorithms did not perform so well. As an example, the ROS *octomap_mapping* package used for obstacle detection is able to provide ground plane detection. However, it does not discern the sheet correctly due to the small height difference.

The division of the sheet into work zones is designed for four different orientations of the robot: 0, 90, 180 and 270º, required to cover all areas of the sheet. The division of the defect according to working zones is calculated hierarchically, giving more priority to the 0° area and lower priority to the 270º. This causes that, in some defects, the repair path is subdivided into more orientations than necessary. We are working on minimizing the number of orientations needed for each defect considering a greedy algorithm that give priority to the orientation that produces the largest area.

Also, other orientations than the default ones can be used. We are working on maximizing the length of the trajectories by adjusting the orientation to the defect principal axis of inertia. But it is not always possible to repair a defect with a single orientation, for example, defects that extends to two opposite sides of the sheet.

V. References

[1] H. M. La *et al.*, 'Autonomous robotic system for high-efficiency non-destructive bridge deck inspection and evaluation', in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2013, pp. 1053–1058. doi: 10.1109/CoASE.2013.6653886.

[2] Z. Gui and H. Li, 'Automated defect detection and visualization for the robotic airport runway inspection', *IEEE Access*, vol. 8, pp. 76100–76107, 2020, doi: 10.1109/ACCESS.2020.2986483.

[3] R. Lim, H. La, and W. Sheng, 'A robotic crack inspection and mapping system for bridge deck maintenance', *Automation Science and Engineering, IEEE Transactions on*, vol. 11, pp. 367–378, Apr. 2014, doi: 10.1109/TASE.2013.2294687.

[4] D. Deng, J. W. Polden, J. Dong, and P. Y. Tao, 'Sensor guided robot path generation for surface repair tasks on a large-scale buoyancy module', *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 636–645, Apr. 2018, doi: 10.1109/TMECH.2018.2797177.

[5] E. Galceran and M. Carreras, 'A survey on coverage path planning for robotics', *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013, doi: 10.1016/j.robot.2013.09.004.

[6] Y. Gabriely and E. Rimon, 'Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot', in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, May 2002, vol. 1, pp. 954–960 vol.1. doi: 10.1109/ROBOT.2002.1013479.

[7] S. X. Yang and C. Luo, 'A neural network approach to complete coverage path planning', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 718–724, Feb. 2004, doi: 10.1109/TSMCB.2003.811769.

[8] B. Chazelle, 'Approximation and decomposition of shapes', in *Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics*, J. T. Schwartz and C. K. Yap, Eds. Lawrence Erlbaum Associates, 1987, pp. 145–185. [Online]. Available: https://www.cs.princeton.edu/ chazelle/pubs.html

[9] H. Choset and P. Pignon, 'Coverage path planning: the boustrophedon cellular decomposition', in *Field and Service Robotics*, A. Zelinsky, Ed. London: Springer London, 1998, pp. 203–209. doi: 10.1007/978-1-4471-1273-0_32.

[10] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, 'Morse Decompositions for Coverage Tasks', *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 331–344, Apr. 2002, doi: 10.1177/027836402320556359.

[11] S. C. Wong and B. A. MacDonald, 'Complete coverage by mobile robots using slice decomposition based on natural landmarks', in *PRICAI 2004: Trends in Artificial Intelligence*, vol. 3157, C. Zhang, H. W. Guesgen, and W.-K. Yeap, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 683–692. doi: 10.1007/978-3-540-28633-2_72.

[12] C. N. Macleod, G. Dobie, S. G. Pierce, R. Summan, and M. Morozov, 'Machining-Based Coverage Path Planning for Automated Structural Inspection', *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 202–213, Jan. 2018, doi: 10.1109/TASE.2016.2601880.

[13] G. Grisettiyz, C. Stachniss, and W. Burgard, 'Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling', in *Proceedings of the 2005 IEEE international conference on robotics and automation*, 2005, pp. 2432–2437.

[14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, 'OctoMap: an efficient probabilistic 3D mapping framework based on octrees', *Auton Robot*, vol. 34, no. 3, pp. 189–206, Apr. 2013, doi: 10.1007/s10514-012-9321-0.

[15] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, 'Generation of fiducial marker dictionaries using Mixed Integer Linear Programming', *Pattern Recognition*, vol. 51, pp. 481–491, Mar. 2016, doi: 10.1016/j.patcog.2015.09.023.

[16] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, 'Speeded up detection of squared fiducial markers', *Image and Vision Computing*, vol. 76, pp. 38–47, Aug. 2018, doi: 10.1016/j.imavis.2018.05.004.

[17] D. H. Douglas and T. Peucker, 'Algorithms for the reduction of the number of points required to represent a digitized line or its caricature', 1973, doi: 10.3138/FM57-6770-U75U-7727.

[18] Á. Fernández García, S. Roos Hoefgeest Toribio, I. Álvarez García, and R. C. González de los Reyes, 'Algoritmo de generación de trayectorias en el interior de chapas para la subsanación de defectos', in *Libro de actas*, Aug. 2019, pp. 702–709. doi: 10.17979/spudc.9788497497169.702.

[19] Chazelle, B., 'Approximation and decomposition of shapes.', in *Advances in Robotics*, 1987, pp. 145–185.

[20] M. Quigley *et al.*, 'ROS: an open-source Robot Operating System', in *ICRA workshop on open source software*, 2009, vol. 3, no. 3.2, p. 5.

[21] G. Bradski, 'The opencv library', *Dr Dobb's Journal of Software Tools*, vol. 25, pp. 120–125, 2000.

[22] P. Schillinger, S. Kohlbrecher, and O. von Stryk, 'Human-robot collaborative high-level control with application to rescue robotics', in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2796–2802. doi: 10.1109/ICRA.2016.7487442.