Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 10, No. 4, December 2024, pp. 942-959
ISSN: 2338-3070, DOI: 10.26555/jiteki.v10i4.29806 

# Optimizing Cleaning Path for Coal Dust Removal Using Dual Stage Tracking Method

Ira Kumalasari[1], Harits Ar Rosyid[1], Siti Sendari[1], Norrima Binti Mokhtar[2], Samsul Setumin[3]
[1]Universitas Negeri Malang, Jl. Semarang No.5, Malang 65141, Indonesia
[2]University of Malaya, Universiti Malaya, 50603 Kuala Lumpur, Wilayah Persekutuan Kuala Lumpur, Malaysia
[3]Universiti Teknologi MARA (UiTM), Cawangan Pulau Pinang Kampus Permatang Pauh, 13500 Permatang Pauh, Pulau Pinang, Malaysia

**ARTICLE INFO**

**Article history:**

Received August 15, 2024
Revised January 15, 2025
Accepted January 25, 2025

**Keywords:**

Disaster Mitigation;
Cleaning Robot;
Single stage BFS;
Single Stage A*;
Dual Stage A*

**ABSTRACT**

Manual disaster mitigation at the Java Bali power plant, particularly related to fire risks from coal dust during electricity production, often requires halting operations, leading to significant revenue loss and power outages. This study aims to address this issue by proposing an automated solution to clean coal dust without interrupting production, utilizing a dual-stage tracking method for robot-assisted coal dust cleaning. The research contributes by developing a dual-stage A* algorithm that optimizes robot movements for cleaning tasks in power plant environments, outperforming single-stage BFS and single-stage A* algorithms. The research is divided into two phases: object detection and robot motion path selection. The dual-stage A* algorithm is compared against single-stage BFS and single-stage A* methods through a series of experiments evaluating their efficiency and effectiveness. The dual-stage A* method demonstrates superior performance in terms of path optimization, reducing cleaning time, and improving operational safety. Specifically, the dual-stage A* algorithm reduces energy consumption by 169 units and grid traversal by 84 units compared to single-stage methods, ensuring thorough dust removal while minimizing fire hazards. The dual-stage A* algorithm proves to be the optimal solution for coal dust cleaning in power plants, allowing for safe, continuous operation without the need for production halts. Future work should focus on addressing implementation costs and technical constraints to enhance real-world applicability.



**Corresponding Author**:

Harits Ar Rosyid, Universitas Negeri Malang, Jl. Semarang No.5, Malang 65141, Indonesia
Email: harits.ar.ft@um.ac.id

## 1. INTRODUCTION

Manual disaster mitigation at the Java-Bali power plant, particularly related to fire risks from coal dust during electricity production, often requires halting operations. This leads to significant revenue loss and power outages, highlighting the need for an automated solution to clean coal dust without interrupting production. Coal dust poses severe risks, including self-combustion in coal barges, coal yards, hot surfaces, and crusher bodies [1], [2]. Current manual cleaning processes not only pose health and safety risks to workers but also necessitate production halts in the event of a disaster, leading to substantial revenue loss and negatively impacting consumers [3], [4]. Therefore, an automated solution for continuous coal dust cleaning is urgently needed.

The frequent manual cleaning required in coal-based power plants, especially in critical areas such as the tripper area, poses substantial health risks and operational inefficiencies[5], [6]. Despite advancements in robotic cleaning technologies, there remains a gap in integrating real-time dust detection with optimized robot motion planning to maintain continuous operations. Previous studies have explored various aspects of robotic

cleaning, including camera-based object detection and navigation algorithms[7]. However, these studies have not fully addressed the integration of dust detection and autonomous cleaning in dynamic power plant environments. Recent advancements in single-stage BFS and A* algorithms have shown potential, but their application in complex industrial settings remains limited [8], [9].

Existing research on robotic cleaning in industrial environments has primarily focused on static object detection and pathfinding. For instance, contour detection techniques have been employed for object recognition in various settings [10], [11], [12]. However, these methods often lack the robustness needed for real-time dust detection and cleaning in power plants. Additionally, while algorithms such as BFS and A* have been utilized for path optimization, their performance in dynamic and dusty environments has not been thoroughly evaluated [13].

This study bridges the gap by integrating a camera-based dust detection system with a dual-stage A* algorithm for optimized path planning. Unlike traditional methods, this approach allows for continuous monitoring and cleaning of coal dust, reducing the risk of fire hazards and enhancing overall operational efficiency. The primary objective of this research is to develop and evaluate a dual-stage tracking method that enables autonomous robots to clean coal dust continuously in power plants. Specifically, the study aims to:

1. Integrate real-time dust detection with robot motion planning.
2. Compare the performance of the dual-stage A* algorithm against single-stage BFS and A* algorithms.
3. Assess the practical implications of the proposed method for power plant operations, including reduced downtime, improved safety, and cost savings.

Implementing an automated cleaning system in a power plant environment involves several challenges, including handling variable lighting conditions for dust detection and ensuring reliable robot navigation in a dynamic setting. These challenges necessitate robust algorithms and real-time data processing capabilities, which this study addresses through the proposed dual-stage A* algorithm. The findings of this study have significant practical implications for disaster mitigation in power plants. By enabling continuous coal dust cleaning, the proposed method can reduce operational downtime, enhance worker safety, and lower maintenance costs. Moreover, the approach can be scaled and adapted to other industrial settings where dust accumulation poses similar risks. The integration of automated systems in industrial environments raises important ethical and safety considerations, such as the potential displacement of workers and system reliability. This study acknowledges these concerns and emphasizes the importance of ensuring that automated solutions complement human labor, enhancing overall safety and efficiency without compromising employment.

The research contribution is:

1. Developing a dual-stage A* algorithm for optimized cleaning path planning in power plants.
2. Demonstrating the practical benefits of integrating real-time dust detection with autonomous cleaning to improve operational safety and efficiency.

By addressing the identified research gap, this study advances the state of the art in robotic cleaning technologies and provides a viable solution for continuous coal dust cleaning in power plants

## 2. METHODS

This research focuses on the simulation of a dual-stage tracking method for optimizing cleaning tasks in power plants, specifically targeting the Java-Bali power plant. The study has not yet been integrated with an actual cleaning robot but lays the groundwork for future implementation. The methodology involves detailed steps for data collection, preprocessing, object detection, pathfinding algorithms, and heuristic functions to achieve efficient and effective cleaning. The data collection process involved gathering images from the tripper area of the Java-Bali power plant, where coal dust accumulation is most critical. Preprocessing steps included handling missing values, normalization, and outlier detection to ensure data quality. Coal dust detection was performed using the HSV color space, effective in uncertain lighting conditions. Contours of detected dust areas were identified and used as inputs for robot motion simulation. The algorithms evaluated include Greedy, BFS (Breadth-First Search), and A*, with hyperparameter tuning performed using grid and random search methods to optimize performance.

The experimental setup was conducted in a virtual environment simulating the conditions of the Java-Bali power plant. The simulation track was designed with dimensions at a scale of 1:3 compared to the actual tripper area, both in length and width. A single camera was positioned along the width of the track, centered, and at a height scaled to 1:3 of the actual camera height in the plant. This setup ensured that the simulation closely mimicked the real-world environment, providing relevant and accurate data for analysis. Hardware specifications use 11th generation intel i7, 16 GB RAM, and 64-bit. The software uses phyton for simulation.

The real-time performance, scalability, and energy consumption of the algorithms were evaluated using statistical tests, including t-tests. Ethical and safety considerations, such as worker displacement and system

reliability, were also addressed to ensure the responsible deployment of automated systems. This study's findings indicate that the dual-stage A* algorithm significantly reduces energy consumption and grid traversal compared to single-stage methods. Future research should focus on real-world implementation, addressing overfitting, false positives and negatives, and further scalability. By integrating advanced techniques and conducting longitudinal studies, the robustness and efficiency of the algorithm can be enhanced for broader industrial applications.

### 2.1. Object Detection

Detecting coal dust is essential for maintaining cleanliness in power plants. This study utilizes the HSV (Hue, Saturation, Value) color space due to its effectiveness under varying lighting conditions and its operational efficiency. Coal dust appears as solid black in images, corresponding to specific HSV values. The HSV method separates color information from intensity, simplifying processing and reducing system load compared to RGB, where color and brightness are combined[14]. This approach is both computationally efficient and effective for real-time detection in industrial environments [15]. The decision to use HSV is based on its robustness in handling lighting changes, which are common in industrial settings[16]. Unlike RGB, where changes in lighting can alter color appearance, HSV maintains consistent hue values, ensuring reliable detection of black coal dust despite fluctuations in lighting. This robustness is crucial, as the lux levels in both the experimental setup and the actual site were carefully controlled (average lux = 225, minimum lux = 115, maximum lux = 244). These controlled conditions confirmed that HSV is effective, allowing the system to perform in real-time without compromising accuracy. Using HSV for coal dust detection provides the system with lower computational requirements and higher accuracy in variable lighting conditions. Its simplicity and effectiveness make it ideal for real-time applications in power plants. While advanced techniques like YOLO or Mask R-CNN offer high accuracy[17], [18], they are unnecessary for this application and would introduce excessive computational complexity. Therefore, HSV is the best choice for the current experimental setup, considering the controlled lighting and need for efficiency. The implementation involves using the HSV color space to detect coal dust, where the dust appears as solid black with a hue value of 0. This approach is effective even in environments with uncertain lighting. Data collection was performed to obtain contour magnitudes using rule-based logic, and these results were used as input for robot motion planning. The camera used for capturing images had a resolution of 1920×1080 pixels, ensuring high-quality data for accurate detection.

In pseudocode 1, the initial stage was the creation of a database to obtain data from image results in the form of contour classification from the color of detected coal dust [19]. Subsequently, the data was processed by the robot to determine the location of dust to be cleaned. The database was stored in Excel files daily and this was used to determine the trend of dust scattered on the floor daily. Image acquisition was performed using a resolution of 1080p/30fps resolution (1920 pixels×1080 pixels), then normalized to 720p/30fps (maximum 1280 pixels×720 pixels) adjusted to the specifications of the computer screen. This was intended to facilitate more accessible monitoring for the operator. Then, the normalized image was converted into an HSV image with a range of lower (0, 0, 0) and upper (179, 255, 30). Subsequently, the image was converted to HSV, after which contours were identified in the mask [20]. In the frame, a grid measuring 96 x 54 was employed, where one grid box is worth 200 pixels and declared dirty if the black image was ≥100 pixels. Then, the contours are arranged in a list based on their contour area. The sequence numbering of this contour list commences with the smallest contour area and progresses toward the starting point, with the most significant area assigned the highest sequence number. The data can be exported for use in determining the robot's motion path.

**Pseudocode 1: Object Detection**

```
CREATE directory for saving images using current date if not exists
INITIALIZE image counter
WHILE True
    CAPTURE and resize image, convert to HSV
    CREATE mask to detect black color, find contours in mask
    COPY image for drawing
    INITIALIZE data list for Excel
    FOR each contour with area > 1
        CALCULATE dark intensity, contour centroid
        DRAW centroid and contour info on image
        APPEND contour data to list
    SAVE DataFrame to Excel, processed images, and histograms
    WAIT for 15 minutes
    INCREMENT image counter
END
```

### 2.2. Searching Algorithm

In this study, searching algorithms such as BFS, A*, and the Greedy algorithm were employed for path planning. These algorithms differ from machine learning techniques in that they do not rely on data-driven training. Instead, they operate based on predefined rules and heuristics, systematically exploring paths to find the most efficient route. Their use ensures predictable and efficient pathfinding, making them particularly suitable for well-defined environments like power plants, where the complexity of machine learning models is not necessary.

#### 2.2.1. Greedy Algorithm

This algorithm employs local heuristics to make decisions at each step to optimize the regional solution. In many cases, local heuristics in greedy algorithms are relatively straightforward, often using directly available information to make decisions. Greedy algorithm formula refers to equation (1) [21].

**Pseudocode 2: Greedy Algorithm**

```
If length of nearest_coordinates > 1:
    Sort nearest_coordinates based on x value and then x+y value ← equation (1)
Set nearest_coordinate to the first element in nearest_coordinates
```

$$H(n) = \left|\left(x_p - x_s\right) + \left(y_p - y_s\right)\right| \quad (1)$$

$x_s$ and $y_s$ means destination coordinates. $x_p$ and $y_p$ means coordinates at this time.

#### 2.2.2. BFS Algorithm

In path planning using the Breadth-First Search (BFS) algorithm, the main objective was to identify the shortest path between two points in a graph. This is analogous to identifying the most expedient route on a map, where BFS operates by systematically traversing the graph from the initial vertex, evaluating its immediate neighbors initially before progressing to more distant vertices [22]. The BFS process commences with the placement of the initial point of departure into a queue [23]. Subsequently, the Breadth-First Search (BFS) algorithm will examine each neighbor of the initial point, noting the distance, and proceed to traverse from one point to another by tallying the number of steps taken. This process is illustrated in the code snippet 3.

**Pseudocode 3: BFS**

```
def bfs(grid, x, y, visited, end_x, end_y):
    if (x, y) == (end_x, end_y): return -1
    visited[x][y] = True
    for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]: dfs(grid, x + dx, y + dy, visited, end_x, end_y)

def run_bfs(world):
    world.dfs_route, print("dfs start"), dfs(world.graph, world.start_x, world.start_y, world.is_visited, world.end_x,
world.end_y)

best_path_length = len(grid_world.dfs_best_route)
```

#### 2.2.3. A* Algorithm

The A* algorithm is a graph search method designed to efficiently identify the shortest path from a starting point to a target. This algorithm combines two approaches: breadth-first search and weighted search, by using a heuristic function that estimates the lowest cost from the current node to the target node [24]. The A* algorithm calculates the total cost value using the function $f(n) = g(n) + h(n)$, where $g(n)$ is the cost from the start node to node $n$ and $h(n)$ represents the estimated cost from node n to the goal [25]. This ensures that the node most promising to reach the goal at the lowest cost is evaluated first [26]. The detailed information is presented in pseudocode 4.

The operation of A* employs two lists, consisting of an Open List for nodes to be evaluated and a Closed List for nodes that have already been evaluated. Starting from the initial node, the algorithm adds this node to the Open List. Then, the algorithm selects the node with the lowest *f* value from the Open List, moves it to the Closed List, and evaluates all its neighbors. If a neighbor is located in the Closed List, it is ignored, while if it is in the Open List, the algorithm updates the *f* value if the new path is cheaper. This process persists until the goal node is found in the Open List, ensuring that the resulting path is the shortest.

**Pseudocode 4: A***

```
Initialize open_set, parent, g_score, f_score with start
While open_set not empty:
  current = node in open_set with min f_score
  If current is end, return trace path from parent
  Remove current from open_set
  For each neighbor of current:
    new_g = g_score[current] + distance (current, neighbor)
    If new_g < g_score[neighbor]:
      parent[neighbor] = current
      g_score[neighbor] = new_g
      f_score[neighbor] = new_g + heuristic (neighbor, end)
      If neighbor not in open_set, add to open_set
Return None
Function main(file):
   Set coordinates, scan grid, run a_star(grid_world)
```

### 2.3. Shorts Distance Search

The Octile and Manhattan heuristics are commonly used for grid-based pathfinding. The Manhattan heuristic is simpler and works well when movement is restricted to horizontal and vertical directions, but does not account for diagonal movement. The Octile heuristic, which includes diagonal movements, gives more accurate results in grid environments where such movements are allowed, but incurs higher computational costs. Euclidean Distance, another common heuristic, is accurate for continuous spaces but not suitable for grid-based systems. The selection of heuristics in this research uses the Octile and Manhattan Heuristics because this research is grid-based, and these heuristics balance accuracy and computational efficiency.

#### 2.3.1. Octile Heuristic

In the A-star search method, all inputs are analyzed, and all of the obtained routes are evaluated before reaching a search solution. This algorithm is used in two conditions:

a) First condition: To identify the shortest distance between the updated start point and the outer limit of the contour area, which serves as the starting point for the obstacle area to be cleared.
b) Second condition: To identify the shortest distance within the contour.

This study employed the Octile Distance heuristic function, which combined the Manhattan and Chebyshev distance functions. This approach was utilized when movement was permitted in eight directions, with diagonal movement incurring a higher cost (usually sqrt (2)). The octile heuristic formula refers to equation (2) [27]. The octile heuristic is illustrated in pseudocode 5.

**Pseudocode 5: octile heuristic**

```
octile_distance ← equation (2)
distances = [octile_distance(x, current_coordinate) for x in coordinates]
min_distance = min(distances)
END
```

$$x = |x_0 - x_1|$$
$$y = |y_0 - y_1| \quad (2)$$
$$\mathrm{d(x,y)} = \max(|x|,|y|) + \sqrt{(2-1).\min(|x|,|y|)}$$

$d(x,y)$ represents the heuristic value, $x = |x_0 - x_1|$means Calculating the difference between the x-coordinate of the current point and the goal, $y = |y_0 - y_1|$ refers to calculating the difference between the y-coordinate of the current point and the destination, $\max(|x|,|y|)$ is the maximum value of the difference in x and y coordinates. This represents the most significant number of steps required if we only moved diagonally. Meanwhile, $\min(|x|,|y|)$ is the minimum value of the difference in $x$ and $y$ coordinates. This represents the smallest number of required steps if we only moved diagonally. Further, $\sqrt{2}$ covers calculating the square root of 2, which is the length of the diagonal in the grid. $\sqrt{(2-1).\min(|x|,|y|)}$ means multiplies the diagonal length by the minimum value of steps, which represents the number of steps required to move diagonally and the number of steps already calculated horizontally or vertically. $\max(|x|,|y|) + \sqrt{(2-1).\min(|x|,|y|)}$ is

the return of the number of horizontal/vertical steps (the largest between dx and dy) plus the number of diagonal steps, which is the approximate octile distance between the current point and the destination.

### 2.3.2. Manhattan Heuristic

The A* algorithm represents the shortest path-finding algorithm for path planning [28]. The Manhattan heuristic was used to measure the distance between two points on the grid based on horizontal and vertical movements only. Point x is the updated start point, while y covers the destination point obtained from the processing results of the octile distance algorithm in the first condition. The Manhattan heuristic formula refers to equation (3) [29], [30]. Pseudocode 6 illustrates that robot motion between the initial point and the final point along the contour is conducted using A*, with the heuristic employed being Manhattan. The heuristic value can be observed in $f$_score. The total A* value or grid was passed based on the minimum value of $f$_score, which was then added to the g_score value.

**Pseudocode 6: A* with Manhattan heuristic**

f_score[neighbor] = new_g + heuristic (neighbor, end) ← equation (3)

$$d(x,y) = \sum_i^m |x_i + y_i| \tag{3}$$

$d(x,y)$ means point to be executed, $x_i$ means the coordinate of the i-th x point, and $y_i$ means i-th x-is point coordinate.

## 2.4. Path Planning of a Mobile Robot

A laboratory is the site of research or experimentation. It may be a building or a room dedicated to this purpose [31]. When controlling a robot, the selection of the right path becomes a fundamental problem [32]. Consequently, the objective of path selection or planning is to assist the robot in identifying the shortest or most optimal route between two points (Field [33]). In some cases, the optimal path is different from the shortest path [34]. Nevertheless, it is subject to a number of criteria, including the minimization of the number of turns or robot movements without colliding with obstacles [35]. Path selection systems on robots can be categorized into several types, including optimal path selection in robot work areas and robot work area maps [36]. Path selection of robot motion can be performed using single stage and dual stage. Path planning for a single stage represents the robot movement based on the list of contours obtained from object detection. While the dual stage for robot motion is divided into 2 stages, the first stage is grouping contours based on groups and the second stage is completing contours based on the largest group to the smallest group.

### 2.4.1. Path Selection using Single-Stage

In single-stage path selection, the BFS and A* algorithms are compared to find the fastest path and clean dust in the detected contours [37]. The heuristic employed was the Octile distance heuristic for designing contours using the Greedy algorithm. For single-stage BFS, pseudocode two is provided in pseudocode 7.

**Pseudocode 7: Single Stage BFS**

START

// Path Completion
1. Find the nearest contour point to the Start point using Octile Heuristic
2. Calculate the distance from the Start point to the nearest contour point
3. Execute the path using BFS to find the closest path
4. Update the Start point to the nearest contour point
5. Find the nearest point inside the contour using Greedy Algorithm
6. Execute the path using BFS to find the closest path
7. Update the Start point to the last executed point
8. Check if all contour areas are completed
   - If no, repeat from step 1
   - If yes, proceed to STOP

STOP

In pseudocode 7, the robot will start its movement following the contour list obtained from object detection. The shortest point between the destination point and the starting point can be identified from the list using the octile heuristic. Subsequently, the A* algorithm was employed to determine the optimal motion path

between the destination point and the start point. Once the destination point was identified, it was removed from the obstacle list, and the latest updated start point was recorded. Subsequently, the point that must be resolved in the contour employs the greedy algorithm. This process continued to repeat until all the contour lists were resolved and the robot returned to the stop point.

The difference for the single stage A* is in solving the closest path between the updated start point and the destination point at the beginning of the contour obtained from the octile heuristic formula. The difference is presented in pseudocode 8. For solving inside the contour, it is the same as single-stage BFS.

**Pseudocode 8: Single Stage A***

```
3. Use A* (Manhattan Heuristic) to find the shortest path from the Start point to the nearest contour point
6. Execute the path using A* (Manhattan Heuristic)  to find the closest path
```

### 2.4.2. Path Selection using Dual-Stage

Dual-stage path selection represents an advancement from single-stage path selection, which fails to consider the contour area. For dual-stage pathfinding, the contour area serves as the main key [38], [39]. The robot is guided by the largest contour area [40], [41].

In dual-stage path selection, there are two stages for robot motion being utilized, as described in the following,

a) The first stage was used to group contours based on the area obtained from image processing. From the list of contours obtained, it was grouped based on formula (1).

**Pseudocode 9: groups contour**

```
Read Excel file into a pandas DataFrame df
Remove commas from 'Area' column and round the values
Sort df based on 'Area' column
Group data by 'Label' and 'Contour Number' into grouped_data
Flatten grouped_data into flattened_data
Initialize grid_world object with dimensions m=96 and n=54
Initialize an empty list obstacles
Initialize an empty set checked
For each obstacle in flattened_data:
    Add obstacle to grid_world
    If obstacle is not in checked, add it to obstacles list
Initialize an empty list obstacle_groups
For each obstacle in obstacles, perform flood fill on grid_world, update checked, and append to obstacle_groups
Remove empty obstacle groups from obstacle_groups
Sort obstacle_groups based on their length (number of cells) ← Equation (4)
Combine obstacle_groups based on their length into obstacle_groups_combined
```

$$O_n = \frac{Tg}{2^n} \tag{4}$$

Equation (4) shows the pheromone, with n representing the value of each division of the grouping, Tg is the total grid value of the black-colored grid, and On is the nth barrier group obtained from the total grid divided by 2 for each grouping. Pheromone 1 results are summarized in Table 1. Table 1 illustrates the utilization of six groups for contour grouping, which facilitates the robot's ability to discern which portion of the contour requires immediate attention. The sequence of groups that must be addressed initially is 1-2-3-4-5-6.

**Table 1.** Obstacle grub division is based on the number of grids.

| *n* | *number of Grids ($O_n$)* | *Group* | *Range Group* |
|---|---|---|---|
| 0 | 5184 | 0 | - |
| 1 | 2592 | 0 | - |
| 2 | 1296 | 1 | grid > 1296 |
| 3 | 648 | 2 | 648 < grid ≥ 1296 |
| 4 | 324 | 3 | 324 < grid ≥ 648 |
| 5 | 162 | 4 | 81 < grid ≥ 324 |
| 6 | 81 | 5 | 40 < grid ≥ 81 |
| 7 | 40,5 | 6 | 0 < grid ≥ 40 |

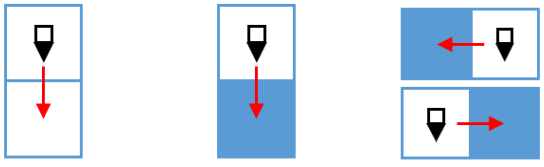
b) The second stage was the selection of the largest contour area to be completed by the greedy algorithm. The starting point of the contour was determined by comparing the same contour in the grub using the A* algorithm (Octile heuristic function) and followed by the selection of the shortest path at the beginning of the path to the starting point of the contour using the A* algorithm (Manhattan heuristic). Completion of the area within the contour was determined using A* (Octile distance function), and solving within the contour was performed using the Greedy algorithm [42], [43], which can be seen in Pseudocode 10.

**Pseudocode 10: Dual stage A***

```
START

// Stage 1: Grouping Contours
1. Find the largest contour group
2. Get the closest contour point to the Start point using Octile Heuristic
3. Compare which contour has the closest point to the Start point
4. If multiple contours are the same, find the nearest path from the contour point to the Start point using A*
(Manhattan Heuristic)
5. Update the Start point to the closest contour point

// Stage 2: Path Completion
6. Find the nearest path within the contour using Octile Heuristic
7. Get the closest point from the contour start point to the point to be executed
8. Execute the path using A* (Manhattan Heuristic) to find the closest path
9. Use Greedy Algorithm to complete the path within the contour
10. Execute the path using A* (Manhattan Heuristic) to find the closest path
11. Update the Start point to the last executed point
12. Check if all contour areas are completed
   - If no, repeat Stage 2
   - If yes, proceed to STOP

STOP
```

## 2.5. Energy Consumption

Energy consumption is a critical factor in the operation of mobile robots, particularly in tasks such as cleaning where the robot must operate efficiently to conserve energy and complete its tasks effectively. This section details the methodology used to simulate and analyze the energy consumption of the robot during its operation. As the research is based on simulation, energy consumption was evaluated by estimating motor power usage and vacuuming under different conditions. The movement of the robot for single-stage and dual-stage requires energy consumption [44]. The division of energy into three states is presented in Fig. 1. Where 2t represents the energy consumption for the robot moving forward without sucking coal dust, 3t is the energy consumption for the robot moving forward and vacuuming coal dust, and 4t is the energy consumption for the robot turning left and right and performing vacuuming.



(a) (b) (c)

**Fig. 1.** (a) energy consumption for 2t, (b) energy consumption for 3t, (b) energy consumption for 4t

## 2.6. Evaluation using t-test

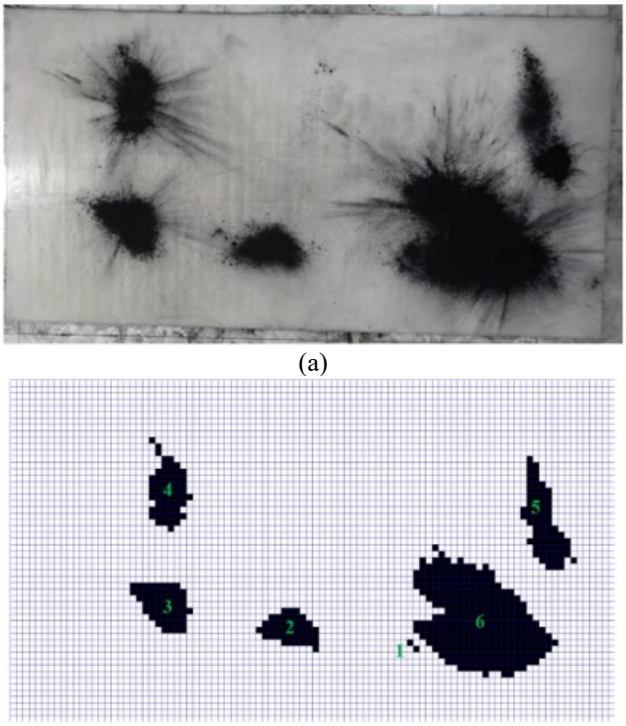
T-test in path planning involved systematically evaluating and comparing the performance of three path planning algorithms (single stage BFS, single stage A*, and dual stage A*) in finding optimal paths in complex environments [45]. This testing aims to ascertain the efficacy of these algorithms in identifying the right path efficiently and according to the set criteria. The path planning test included the Environment, Test Scenario, and Evaluation Metrics [46]. The environment can be a grid, two-dimensional space, or any other simulated environment that reflects the conditions encountered in practical applications [47]. The test scenario included the start point, endpoint, and destination location in contours that must be traversed [48]. Evaluation metrics were used to measure the performance of the path-planning algorithm. Some common metrics used are path cost, execution time, and path quality [49]. Path cost (C) represents the total cost of the path taken by the robot,

calculated as the sum of the costs of all segments in the path, considering factors such as distance traveled and energy consumed [50]. Execution time (T) measures the total time taken by the algorithm to compute the path from the start to the goal, indicating the algorithm's efficiency[51]. Path quality (Q) evaluates the smoothness and efficiency of the path, considering factors such as the number of turns and deviations from the optimal path. These metrics were collected from multiple simulation runs for each algorithm, providing a robust dataset for statistical analysis. A series of t-tests were conducted to compare the effectiveness of different approaches, providing a comprehensive assessment of the algorithms' performance in various scenarios.

## 3. RESULTS AND DISCUSSION

### 3.1. Object Detection

Data collection was carried out eight times. The coal dust utilized in each experiment weighed 100 grams. The distance between the tripper and the floor was 1m, with a random distribution area. Fig. 2 shows the results of object detection from the camera. In accordance with the method used, a grid containing dust less than 100 pixels is eliminated. The contour list is numbered based on the obstacle group, starting from the smallest obstacle group value to the largest obstacle group. The contour area of each tested image is illustrated in Fig. 3.




(a)

(b)

**Fig. 2.** (a) Camera shot results in test case #5, (b) Object detection results in test case #5

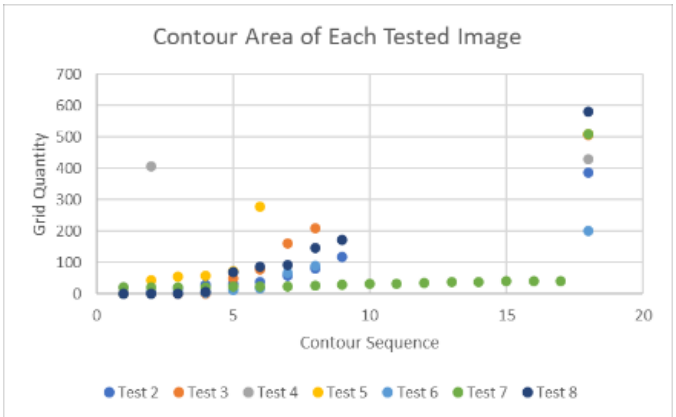



**Fig. 3.** Contour area graph of each tested image

The distribution of the contour area obtained from the random test results demonstrates that eight tests were carried out. The largest contour area is in group 5, with a total grid area of 279. The lowest is in group 6, with a total grid area of 2. The test images exhibit the most significant amount of area in test case #7 and the least in test case #3.

### 3.2. Energy and Accuracy Used Using 3 Modellings

#### 3.2.1. Single-stage BFS

Fig. 4 shows one example of testing the test data for test case #5, wherein the robot's motion starts from path 4-3-2-1-6-5, with a total energy required of 1137981. In Fig. 4, all grids are signified by the color orange, indicating that the robot has passed all grids, in accordance with the BFS theory where all grids will be passed until the destination point is found. The robot moves from the right side to the left. If the robot is on the x-axis of grid number 96, the robot will move rows and start from grid number 96 until it finds the destination point and snakes. This process will persist until all searches for updated start paths to the contour are completed.

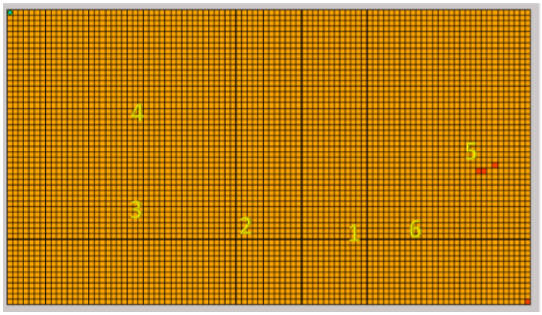



**Fig. 4.** Robot motion in test case #5

#### 3.2.2. Single-stage A*

Fig. 5 presents one example of testing the test data of test case #5, where the robot's motion starts from path 4-3-2-1-6-5, with a total energy requirement of 2147.

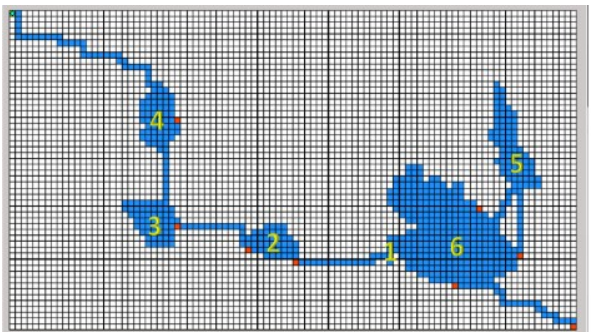



**Fig. 5.** Robot motion in test case #5

#### 3.2.3. Dual-stage A*

Fig. 6 shows an example of testing the test data of test case #5, where the robot's motion starts from path 6-5-2-3-4-1 and has a total energy requirement of 2496.

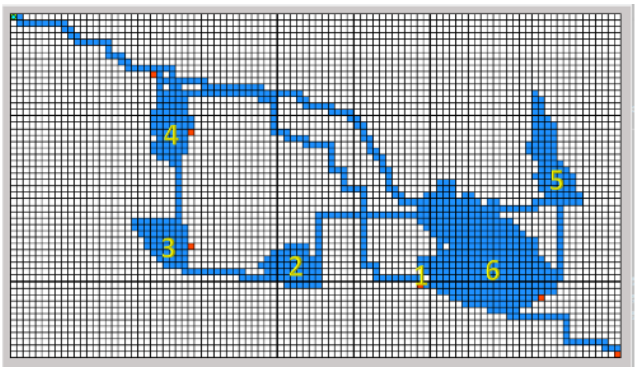



**Fig. 6.** Robot motion in test case #5

### 3.3. Comparison between Single Stage and Dual Stage

The grid completion was garnered from the average ratio between the number of grids passed in each algorithm and the number of grids to complete. As illustrated in Fig. 7, single-stage BFS modeling is markedly ineffective due to the prolonged time required for completion. This is evidenced by the average number of grids that must be completed, which is approximately 687 times. In the case of single-stage A*, the grid completion is approximately 1.6 times the grid that must be completed. In contrast, for dual-stage A*, the grid completion is around 1.7 times the grid to be solved. Single-stage A* is the most effective pathfinding method, followed by dual-stage A* and single-stage BFS. For instance, single-stage A* requires approximately 0.1 fewer times compared to dual-stage A*.

As illustrated in Fig. 7 and Fig.8, the greater number of passed grids represents higher required energy. This is further substantiated in Fig. 9, which depicts that the single-stage BFS algorithm necessitates the highest energy consumption. Conversely, the single-stage A* algorithm exhibits the lowest energy consumption. The average difference in energy consumption between dual-stage A* and single-stage A* is 169.

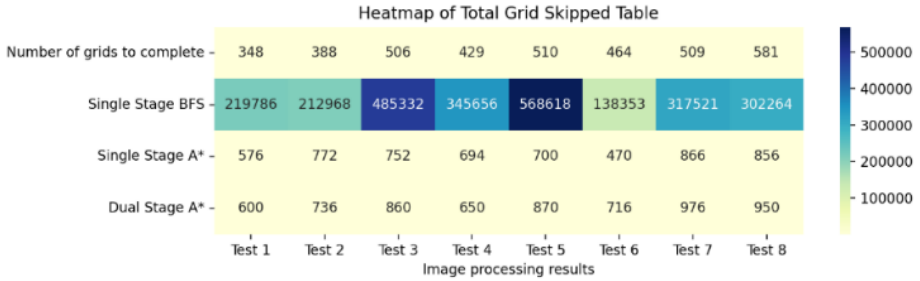



**Fig. 7.** Graph of settlement results using single-stage and dual-stage

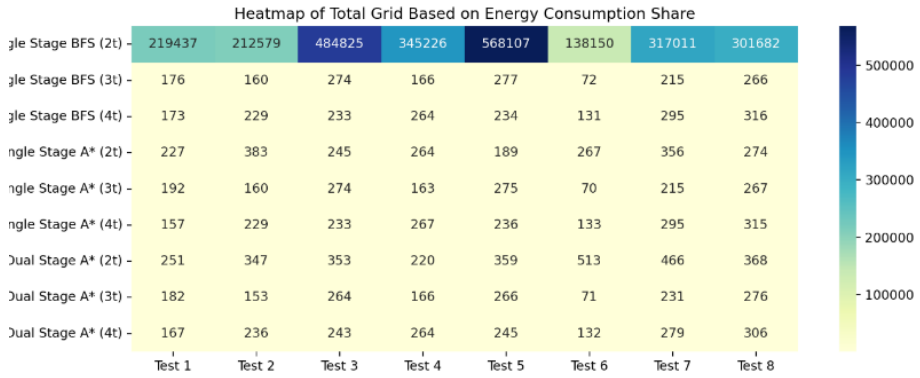



**Fig. 7.** Graph of total grid based on energy consumption share

When the robot moves to a grid with no dust (2t), all three algorithms transverse through a larger number of grids than when the robot moves to a grid with dust (3t and 4t). Single stage BFS (2t) has a larger total grid than single stage A* (2t) and dual stage A* (2t). For single stage A* (2t) and dual stage A* (2t), the pathfinding is nearly identical and more effective than single-stage BFS (2t). For path completion within the contour, all three are almost equally efficient.

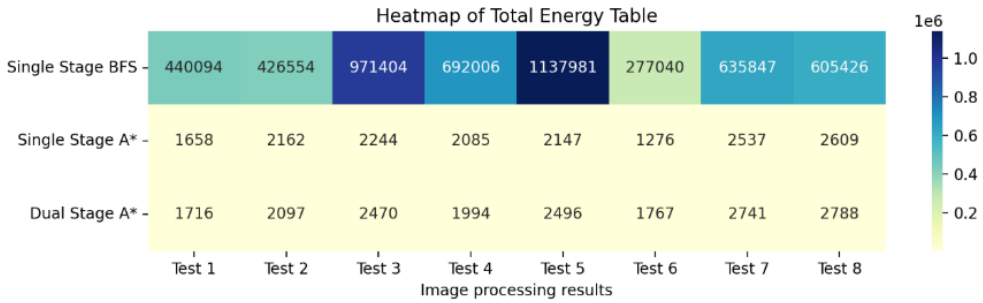



**Fig. 8.** Total energy graph

**Table 2.** Performance of the three Algorithms

| Image processing results | Number of grids to complete | Performance (second) | | |
|---|---|---|---|---|
| | | ***Single Stage BFS*** | ***Single Stage A**** | ***Dual Stage A**** |
| Test 1 | 348 | 1230 | 29 | 25[a] |
| Test 2 | 388 | 1172 | 36 | 26[a] |
| Test 3 | 506 | 3870 | 29[a] | 43 |
| Test 4 | 429 | 4051 | 30 | 29[a] |
| Test 5 | 510 | 4105 | 44 | 35[a] |
| Test 6 | 464 | 366 | 14[a] | 20 |
| Test 7 | 509 | 1515 | 26[a] | 28 |
| Test 8 | 581 | 1602 | 25[a] | 30 |

[a]best performance

The real time performance measured is in simulating the motion of the robor from the start point to the stop point, without comparing with the actual robor motion in the field. As illustrated in Table 2, the performance of the three algorithms shows that Single Stage A*'s computation time is almost similar as Dual Stage A*'s, while BFS requires much longer computation time. Single Stage A* demonstrates superior performance in four test images, while Dual Stage A* performs best in four other test images.

Fig. 10 shows a test case of dual-stage A* performing best compared to single-stage A*. The test results demonstrate that dual-stage A* exhibits superior performance when the contour group difference between contours is pronounced (test cases #1, #2, and #4) and the largest contour grub is situated in grub 4 (test cases #1, #4, and #5).

For the test case Fig. 11, the optimal performance of single stage A* is contingent upon the proximity of the contour group. To illustrate, the group identified within the processed object detection between group 4 and group 5 (test case #3, test case #6, and test case #8), and in instances where the number of contours is considerable (test case #7), will yield superior results when the aforementioned proximity is maintained.

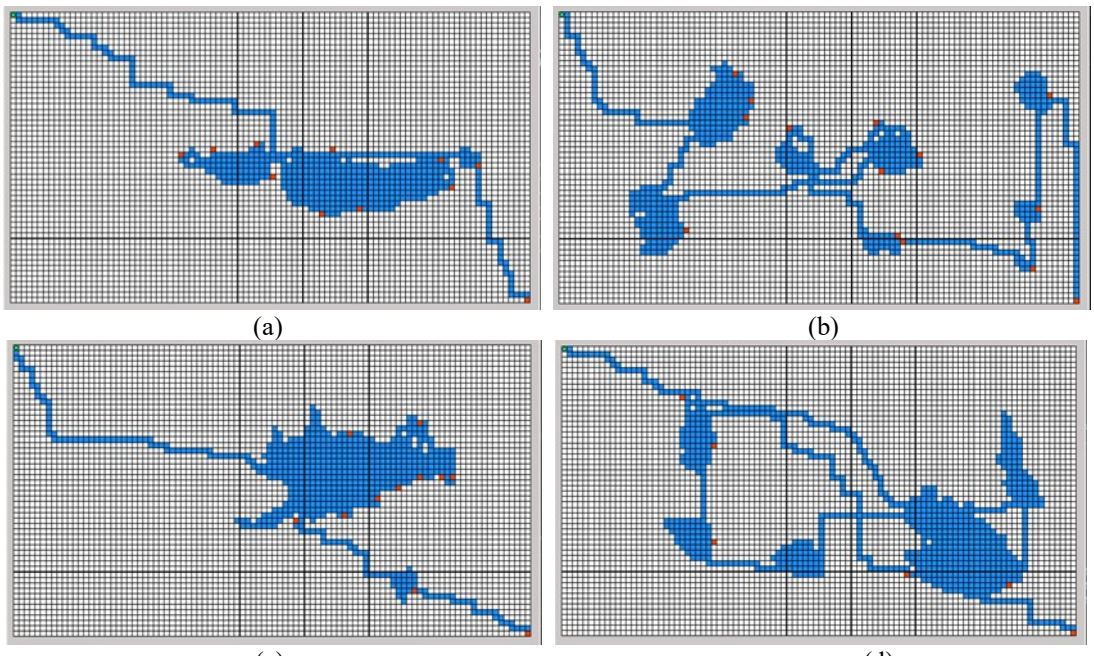
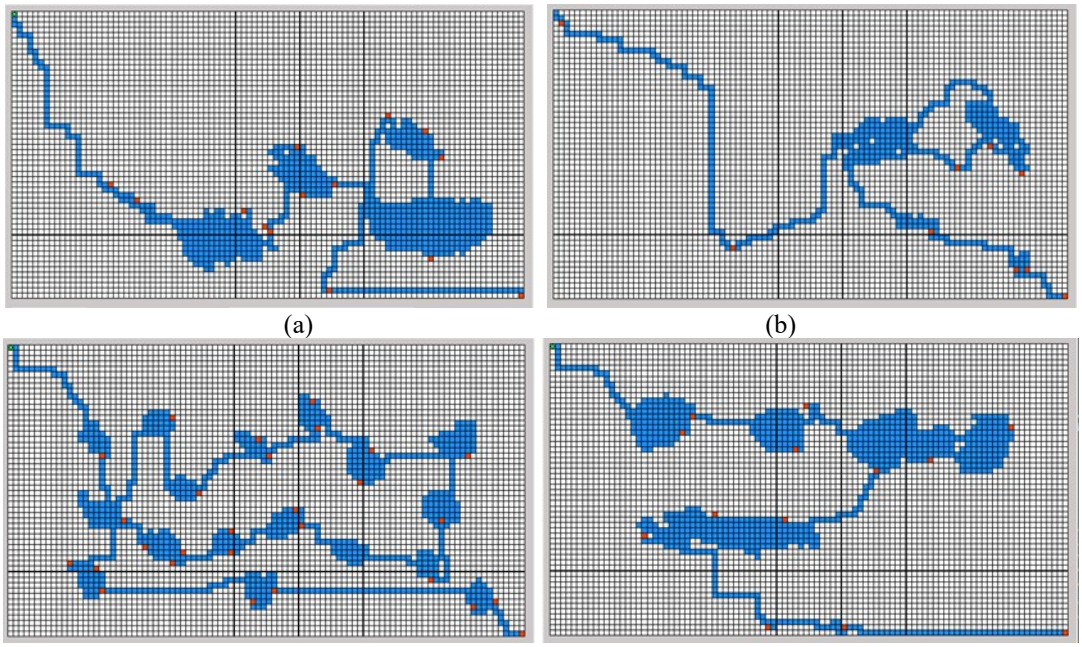



**Fig. 9.** (a) test case #1, (b) test case #2, (c) test case #4, (d) test case #5

Table 3 presents that the Single Stage BFS completion and Single Stage A* trajectories are the same. Both algorithms prioritize the removal of coal dust in the vicinity of the initial point of departure and the updated starting point, effectively disregarding the surrounding contour area. Conversely, Dual Stage A* addresses the largest contour first and progresses to the most minute contour. An illustrative representation of the traversed path for the test can be observed in Fig. 4, Fig. 5, and Fig. 6.



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

**Fig. 10.** (a) test case #3, (b) test case #6, (c) test case #7, (d) test case #8.

**Table 3.** Path trajectory of each test

| Image processing results | Trajectory | | |
|---|---|---|---|
| | ***Single Stage BFS*** | ***Single Stage A**** | ***Dual Stage A**** |
| Test 1 | 2-3-1 | 2-3-1 | 3-2-1 |
| Test 2 | 9-1-5-7-4-2-3-6-8 | 9-1-5-7-4-2-3-6-8 | 9-8-7-5-1-4-2-3-6 |
| Test 3 | 7-2-1-6-8-5-4-3 | 7-2-1-6-8-5-4-3 | 8-7-6-5-4-1-2-3 |
| Test 4 | 2-1 | 2-1 | 2-1 |
| Test 5 | 4-3-2-1-6-5 | 4-3-2-1-6-5 | 6-5-2-3-4-1 |
| Test 6 | 1-2-8-6-3-7-4-5 | 1-2-8-3-7-4-6-5 | 8-7-3-6-5-4-2-1 |
| Test 7 | 3-13-14-6-11-1-2-15-17-16-9-4-5-8-10-12-7 | 3-13-14-6-11-1-2-15-17-16-9-4-5-8-10-12-7 | 16-17-15-2-7-1-11-12-6-14-5-4-9-8-3-13-10 |
| Test 8 | 7-5-1-8-6-9-2-3-4 | 7-5-1-8-6-9-2-3-4 | 9-8-6-7-5-1-3-4-2 |

### 3.4. Comparison between Single Stage and Dual Stage evaluation using t-test

From Table 4, the results of the T-test analysis indicate a significant difference between the Single Stage BFS algorithm and the Single Stage A* and Dual Stage A* algorithms in terms of the total energy used and the number of grids traversed. Single Stage BFS has a mean total energy of 648294 with a standard deviation of 287462.3, which is much higher than Single Stage A* with a mean of 2089.75 and a standard deviation of 439.25. The exceedingly small p-value (8.86947E-06) confirms this significant difference. Analogous outcomes are observed when contrasting the Single Stage BFS with Dual Stage A*, where a p-value of 8.89411E-06 substantiates that Single Stage BFS uses much more energy than Dual Stage A*. Therefore, the Single Stage A* and Dual Stage A* algorithms are the most efficacious in terms of energy efficiency.

Table 5 illustrates that the Single Stage BFS algorithm traverses a significantly greater number of grids than both Single Stage A* and Dual Stage A*. The mean total grid traversed by the Single Stage BFS algorithm is 323,812.25, with a standard deviation of 143,677.51. In comparison, the mean traversed grids for Single Stage A* and Dual Stage A* are 710.75 and 794.75, respectively. The p-values of 8.83449E-06 and 8.85892E-06 for the comparisons with Single Stage A* and Dual Stage A* indicate a statistically significant difference. However, the comparison between Single Stage A* and Dual Stage A* indicates no significant difference in the number of grids traversed, with a p-value of 0.120079071. With respect to the number of traversed grids, the Single Stage A* and Dual Stage A* algorithms demonstrate superior performance relative to Single Stage BFS.

As presented in Table 6, the Single Stage BFS algorithm exhibits significantly inferior performance compared to both Single Stage A* and Dual Stage A*. With a mean performance of 2238.875 and a standard deviation of 1512.81, Single Stage BFS performs much less than Single Stage A* with a mean of 29.125 and Dual Stage A* with a mean of 29.5. The relatively small p-values (0.000508904 and 0.000509577) serve to

confirm the significance of this observed difference. Nevertheless, no significant distinction in performance is evident between Single Stage A* and Dual Stage A*, as indicated by a p-value of 0.462622907. In terms of performance, both Single Stage A* and Dual Stage A* are the most effective, demonstrating that these two algorithms are more efficient and effective than Single Stage BFS.

**Table 4.** The result of the t-test total energy

| | Single Stage Bfs | Single Stage A* | Dual Stage A* |
|---|---|---|---|
| Mean | 648294.00 | 2089.75 | 2258.63 |
| Deviasi | 287462.30 | 439.25 | 421.94 |
| P-value | - | 8.86947E-06 | |
| P-value | - | | 8.89411E-06 |
| P-value | | - | 0.222987722 |

Alpha (α)=0.05

**Table 5.** The result of the t-test total passed grid

| | Single Stage Bfs | Single Stage A* | Dual Stage A* |
|---|---|---|---|
| Mean | 323812.25 | 710.75 | 794.75 |
| Deviasi | 143677.51 | 134.78 | 139.08 |
| P-value | - | 8.83449E-06 | |
| P-value | - | | 8.85892E-06 |
| P-value | | - | 0.120079071 |

**Table 6.** The result of the t-test performance

| | Single Stage Bfs | Single Stage A* | Dual Stage A* |
|---|---|---|---|
| Mean | 2238.875 | 29.125 | 29.5 |
| Deviasi | 1512.81 | 8.66 | 6.95 |
| P-value | - | 0.000508904 | |
| P-value | - | | 0.000509577 |
| P-value | | - | 0.462622907 |

The results of the testing demonstrate that the Dual Stage A* algorithm is the best among those tested, demonstrating superior efficiency and performance in various measured aspects. The Dual Stage A* algorithm was observed to exhibit superior energy efficiency compared to Single Stage BFS, with a notable discrepancy (p-value of 8.89411E-06). Furthermore, this algorithm traverses a significantly smaller number of grids than Single Stage BFS, with a p-value of 8.85892E-06. The results demonstrate that Dual Stage A* outperforms Single Stage BFS in terms of mean performance, with a significantly lower value (29.5 compared to 2238.875) and a very small p-value (0.000509577). Although there is no significant difference in energy usage and the number of grids traversed between Single Stage A* and Dual Stage A*, the performance of Dual Stage A* remains slightly superior, with a lower standard deviation (6.95 compared to 8.66), indicating greater consistency.

This research addresses the gap between the detection of coal dust using camera-based robots and the optimization of cleaning through advanced motion planning algorithms. By effectively integrating these components, the study enhances operational efficiency and safety and lays the groundwork for predictive maintenance strategies in industrial cleaning applications. Future developments could focus on real-time adaptation to dynamically changing dust patterns and further refinement of motion planning algorithms to maximize efficiency in various environmental conditions.

This research offers a valuable contribution to the advancement of robotics technology for automated cleaning applications in industrial environments. By focusing on the integration between object detection using cameras and navigation algorithms such as Single Stage A* and Dual Stage A*, this research highlights the potential for improving operational efficiency and workplace safety. The findings demonstrate that using the Dual Stage A* navigation algorithm can reduce the time to complete the robot's journey and optimize the required energy consumption. This has positive implications for reducing operational costs and increasing productivity in industrial sectors that require regular cleaning of hazardous or dusty areas. However, the discussion should prioritize enhancing the integration between object detection and robot navigation to provide a more holistic and effective solution in practical applications.

On the other hand, this research also identified deficiencies that must be addressed in developing and implementing this technology. One of the primary challenges is the system's reliance on the camera's image quality and object detection accuracy. The system's overall performance may be affected by variability in environmental conditions, such as changes in light or irregular dust patterns. Furthermore, the intricacy of

implementation on an industrial scale necessitates a meticulous approach to risk management and integration with existing infrastructure. Additional consideration should be given to safety, particularly in interactions between robots and human workers in dynamic and potentially hazardous work environments. Ongoing discourse should culminate in initiatives to address these challenges by developing more reliable technologies and integrative solutions that can be more widely adopted across industries.

## 4. CONCLUSION

This study presents a detailed exploration of single-stage and dual-stage path planning algorithms, specifically BFS, A*, and Greedy, for optimizing the navigation of mobile robots in complex environments. The simulation results indicate that while single-stage algorithms, such as BFS and A*, are effective in certain scenarios, the dual-stage approach provides significant improvements in handling larger contour areas by grouping and prioritizing them. This leads to more efficient path planning and better utilization of the robot's capabilities. The evaluation metrics, including path cost, execution time, and path quality, demonstrate the superiority of the dual-stage algorithm in achieving lower path costs and higher path quality compared to single-stage methods.

Future work will focus on real-time performance evaluations and the integration of dynamic obstacle handling to further enhance the practical applicability of the proposed algorithms. Additionally, exploring ensemble methods and incorporating energy-based constraints into the pathfinding algorithms could provide further improvements in robustness and efficiency. The study also acknowledges potential limitations, such as the reliance on simulation data, which may not fully capture real-world complexities. Therefore, subsequent research will involve validating the algorithms in real-world environments to ensure their reliability and effectiveness in practical applications.

## Acknowledgments

I am immensely grateful to the Ministry of Education and to Universitas Negeri Malang for their invaluable guidance and support throughout my study, as well as their generous financial assistance (Master Thesis Research Grant) which has been instrumental in enabling me to undertake this research.

## REFERENCES

[1] D. Dwipayana, I. Garniwa, and H. Herdiansyah, "Sustainability Index of Solar Power Plants in Remote Areas in Indonesia," *Technol. Econ. Smart Grids Sustain. Energy*, vol. 6, no. 1, p. 2, Dec. 2021, https://doi.org/10.1007/s40866-020-00098-0.

[2] H. B. Tambunan *et al.*, "The Challenges and Opportunities of Renewable Energy Source (RES) Penetration in Indonesia: Case Study of Java-Bali Power System," *Energies*, vol. 13, no. 22, p. 5903, Nov. 2020, https://doi.org/10.3390/en13225903.

[3] R. Ramadhan, M. T. Mon, S. Tangparitkul, R. Tansuchat, and D. A. Agustin, "Carbon capture, utilization, and storage in Indonesia: An update on storage capacity, current status, economic viability, and policy," *Energy Geosci.*, vol. 5, no. 4, p. 100335, Oct. 2024, https://doi.org/10.1016/j.engeos.2024.100335.

[4] Md. A. Habib and R. Khan, "Environmental Impacts of Coal-Mining and Coal-Fired Power-Plant Activities in a Developing Country with Global Context," in *Spatial Modeling and Assessment of Environmental Contaminants*, pp. 421–493, 2021, https://doi.org/10.1007/978-3-030-63422-3_24.

[5] A. K. Yadav, "Human health risk assessment in opencast coal mines and coal-fired thermal power plants surrounding area due to inhalation," *Environ. Chall.*, vol. 3, p. 100074, Apr. 2021, https://doi.org/10.1016/j.envc.2021.100074.

[6] B. B. Mandal, S. Bhattacharya, V. D. Manwar, and S. A. Hussain, "Health risk of exposure to noise in coal preparation and mineral processing plants," in *Innovative Exploration Methods for Minerals, Oil, Gas, and Groundwater for Sustainable Development*, pp. 139–157, 2022, https://doi.org/10.1016/B978-0-12-823998-8.00062-4.

[7] D. Lim, J. Kim, and H. Kim, "Efficient robot tracking system using single-image-based object detection and position estimation," *ICT Express*, vol. 10, no. 1, pp. 125–131, Feb. 2024, https://doi.org/10.1016/j.icte.2023.07.009.

[8] S. Rani, D. Ghai, and S. Kumar, "Object detection and recognition using contour based edge detection and fast R-CNN," *Multimed. Tools Appl.*, vol. 81, no. 29, pp. 42183–42207, Dec. 2022, https://doi.org/10.1007/s11042-021-11446-2.

[9] T.-T. Nguyen and C. Vo Duy, "Grasping moving objects with incomplete information in a low-cost robot production line using contour matching based on the Hu moments," *Results Eng.*, vol. 23, p. 102414, Sep. 2024, https://doi.org/10.1016/j.rineng.2024.102414.

[10] I. Cuiral-Zueco, Y. Karayiannidis, and G. López-Nicolás, "Contour Based Object-Compliant Shape Control," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 5164–5171, Aug. 2023, https://doi.org/10.1109/LRA.2023.3292617.

[11] V. S. Panwar, A. Pandey, and Md. E. Hasan, "Design and fabrication of a novel concept-based autonomous controlled solar powered four-wheeled Floor Cleaning Robot for wet and dry surfaces," *Int. J. Inf. Technol.*, vol. 14, no. 4, pp. 1995–2004, Jun. 2022, https://doi.org/10.1007/s41870-022-00893-1.

[12] Z. Yan, S. Schreiberhuber, G. Halmetschlager, T. Duckett, M. Vincze, and N. Bellotto, "Robot perception of static and dynamic objects with an autonomous floor scrubber," *Intell. Serv. Robot.*, vol. 13, no. 3, pp. 403–417, Jul. 2020, https://doi.org/10.1007/s11370-020-00324-9.
[13] M. Wang, J. Xu, J. Zhang, and Y. Cui, "An autonomous navigation method for orchard rows based on a combination of an improved a-star algorithm and SVR," *Precis. Agric.*, vol. 25, no. 3, pp. 1429–1453, Jun. 2024, https://doi.org/10.1007/s11119-024-10118-z.
[14] R. M. A. Eshaq, E. Hu, M. Li, and M. S. Alfarzaeai, "Separation Between Coal and Gangue Based on Infrared Radiation and Visual Extraction of the YCbCr Color Space," *IEEE Access*, vol. 8, pp. 55204–55220, 2020, https://doi.org/10.1109/ACCESS.2020.2981534.
[15] H.-C. Kang, H.-N. Han, H.-C. Bae, M.-G. Kim, J.-Y. Son, and Y.-K. Kim, "HSV Color-Space-Based Automated Object Localization for Robot Grasping without Prior Knowledge," *Appl. Sci.*, vol. 11, no. 16, Art. no. 16, Jan. 2021, https://doi.org/10.3390/app11167593.
[16] X. Chen, Y. Chen, and G. Zhang, "A computer vision algorithm for locating and recognizing traffic signal control light status and countdown time," *J. Intell. Transp. Syst.*, vol. 25, no. 5, pp. 533–546, Sep. 2021, https://doi.org/10.1080/15472450.2021.1871611.
[17] G. Xue, S. Li, P. Hou, S. Gao, and R. Tan, "Research on lightweight Yolo coal gangue detection algorithm based on resnet18 backbone feature network," *Internet Things*, vol. 22, p. 100762, Jul. 2023, https://doi.org/10.1016/j.iot.2023.100762.
[18] X. Wang, S. Wang, Y. Guo, X. Jia, K. Hu, and G. Cheng, "Multi-scale coal and gangue detection in dense state based on improved Mask RCNN," *Measurement*, vol. 221, p. 113467, Nov. 2023, https://doi.org/10.1016/j.measurement.2023.113467.
[19] Z. Wang, D. Li, X. Zheng, and D. Xie, "A Novel Coal Dust Characteristic Extraction to Enable Particle Size Analysis," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021, https://doi.org/10.1109/TIM.2021.3113127.
[20] Z. Zhang *et al.*, "Multi-information online detection of coal quality based on machine vision," *Powder Technol.*, vol. 374, pp. 250–262, Sep. 2020, https://doi.org/10.1016/j.powtec.2020.07.040.
[21] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms," *IEEE Access*, vol. 9, pp. 119310–119342, 2021, https://doi.org/10.1109/ACCESS.2021.3108177.
[22] A. Majumder, "Pathfinding and Navigation," in *Deep Reinforcement Learning in Unity*, Berkeley, CA: Apress, pp. 73–153, 2021, https://doi.org/10.1007/978-1-4842-6503-1_2.
[23] D. D. V. S. Ashish, S. Munjal, M. Mani, and S. Srivastava, "Path Finding Algorithms," in *Emerging Technologies in Data Mining and Information Security*, vol. 1286, pp. 331–338, 2021, https://doi.org/10.1007/978-981-15-9927-9_33.
[24] A. S. Abdel-Rahman, S. Zahran, B. E. Elnaghi, and S. F. Nafea, "Enhanced Hybrid Path Planning Algorithm Based on APF and A-Star," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLVIII-1/W2-2023, pp. 867–873, Dec. 2023, https://doi.org/10.5194/isprs-archives-XLVIII-1-W2-2023-867-2023.
[25] L. Liu, B. Wang, and H. Xu, "Research on Path-Planning Algorithm Integrating Optimization A-Star Algorithm and Artificial Potential Field Method," *Electronics*, vol. 11, no. 22, p. 3660, Nov. 2022, https://doi.org/10.3390/electronics11223660.
[26] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment," *IEEE Access*, vol. 9, pp. 59196–59210, 2021, https://doi.org/10.1109/ACCESS.2021.3070054.
[27] F. Jubair and M. Hawa, "Exploiting Obstacle Geometry to Reduce Search Time in Grid-Based Pathfinding," *Symmetry*, vol. 12, no. 7, p. 1186, Jul. 2020, https://doi.org/10.3390/sym12071186.
[28] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of Autonomous Path Planning Algorithms for Mobile Robots," *Drones*, vol. 7, no. 3, p. 211, Mar. 2023, https://doi.org/10.3390/drones7030211.
[29] B. Yan, T. Chen, X. Zhu, Y. Yue, B. Xu, and K. Shi, "A Comprehensive Survey and Analysis on Path Planning Algorithms and Heuristic Functions," in *Intelligent Computing*, vol. 1228, pp. 581–598, 2020, https://doi.org/10.1007/978-3-030-52249-0_39.
[30] C. Jiang *et al.*, "A Novel Dual-Robot Accurate Calibration Method Using Convex Optimization and Lie Derivative," *IEEE Trans. Robot.*, vol. 40, pp. 960–977, 2024, https://doi.org/10.1109/TRO.2023.3344025.
[31] N. Melenbrink, J. Werfel, and A. Menges, "On-site autonomous construction robots: Towards unsupervised building," *Autom. Constr.*, vol. 119, p. 103312, Nov. 2020, https://doi.org/10.1016/j.autcon.2020.103312.
[32] M. N. A. Wahab, S. Nefti-Meziani, and A. Atyabi, "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?," *Annu. Rev. Control*, vol. 50, pp. 233–252, 2020, https://doi.org/10.1016/j.arcontrol.2020.10.001.
[33] J. A. Abdulsaheb and D. J. Kadhim, "Classical and Heuristic Approaches for Mobile Robot Path Planning: A Survey," *Robotics*, vol. 12, no. 4, p. 93, Jun. 2023, https://doi.org/10.3390/robotics12040093.
[34] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A Survey of Path Planning Algorithms for Mobile Robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, Aug. 2021, https://doi.org/10.3390/vehicles3030027.
[35] B. AlKhlidi, A. T. Abdulsadda, and A. Al Bakri, "Optimal Robotic Path Planning Using Intlligents Search Algorithms," *J. Robot. Control JRC*, vol. 2, no. 6, 2021, https://doi.org/10.18196/jrc.26132.
[36] A. Maoudj and A. Hentout, "Optimal path planning approach based on Q-learning algorithm for mobile robots," *Appl. Soft Comput.*, vol. 97, p. 106796, Dec. 2020, https://doi.org/10.1016/j.asoc.2020.106796.
[37] Á. Madridano, A. Al-Kaff, D. Martín, and A. De La Escalera, "Trajectory planning for multi-robot systems: Methods and applications," *Expert Syst. Appl.*, vol. 173, p. 114660, Jul. 2021, https://doi.org/10.1016/j.eswa.2021.114660.

[38] J. Luo, S. Liu, W. Si, and C. Zeng, "Enhancing Human–Robot Collaboration: Supernumerary Robotic Limbs for Object Balance," *IEEE Trans. Syst. Man Cybern. Syst.*, pp. 1–14, 2024, https://doi.org/10.1109/TSMC.2024.3501389.
[39] S. Sendari, A. N. Afandi, I. A. E. Zaeni, Y. D. Mahandi, K. Hirasawa, and H.-I. Lin, "Exploration of genetic network programming with two-stage reinforcement learning for mobile robot," *TELKOMNIKA Telecommun. Comput. Electron. Control*, vol. 17, no. 3, pp. 1447–1454, 2019, http://doi.org/10.12928/telkomnika.v17i3.12232.
[40] Y. Li, J. Wang, H. Chen, X. Jiang, and Y. Liu, "Object-Aware View Planning for Autonomous 3-D Model Reconstruction of Buildings Using a Mobile Robot," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–15, 2023, https://doi.org/10.1109/TIM.2023.3279424.
[41] S. Sendari *et al.*, "Dual-stage identifying data of arm robot for recognizing and sorting objects with double faces permitted-prohibited area," *J. Adv. Manuf. Technol. JAMT*, vol. 18, no. 1, 2024, [Online]. Available: https://jamt.utem.edu.my/jamt/article/view/6635.
[42] D. Xiang, H. Lin, J. Ouyang, and D. Huang, "Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot," *Sci. Rep.*, vol. 12, no. 1, p. 13273, Aug. 2022, https://doi.org/10.1038/s41598-022-17684-0.
[43] B. Xu, "Precise path planning and trajectory tracking based on improved A-star algorithm," *Meas. Control*, vol. 57, no. 8, pp. 1025–1037, Aug. 2024, https://doi.org/10.1177/00202940241228725.
[44] Z. Tu, F. Fei, and X. Deng, "Untethered Flight of an At-Scale Dual-motor Hummingbird Robot with Bio-inspired Decoupled Wings," *IEEE Robot. Autom. Lett.*, pp. 1–1, 2020, https://doi.org/10.1109/LRA.2020.2974717.
[45] X. Zhang, Z. Liao, L. Ma, and J. Yao, "Hierarchical multistrategy genetic algorithm for integrated process planning and scheduling," *J. Intell. Manuf.*, vol. 33, no. 1, pp. 223–246, Jan. 2022, https://doi.org/10.1007/s10845-020-01659-x.
[46] X. Zhong, J. Tian, H. Hu, and X. Peng, "Hybrid Path Planning Based on Safe A* Algorithm and Adaptive Window Approach for Mobile Robot in Large-Scale Dynamic Environment," *J. Intell. Robot. Syst.*, vol. 99, no. 1, pp. 65–77, Jul. 2020, https://doi.org/10.1007/s10846-019-01112-z.
[47] B. Yang, W. Li, J. Wang, J. Yang, T. Wang, and X. Liu, "A Novel Path Planning Algorithm for Warehouse Robots Based on a Two-Dimensional Grid Model," *IEEE Access*, vol. 8, pp. 80347–80357, 2020, https://doi.org/10.1109/ACCESS.2020.2991076.
[48] C.-Y. Kim and S. Sull, "Grid Graph Reduction for Efficient Shortest Pathfinding," *IEEE Access*, vol. 11, pp. 74263–74276, 2023, https://doi.org/10.1109/ACCESS.2023.3293125.
[49] S. Zeeshan and T. Aized, "Performance Analysis of Path Planning Algorithms for Fruit Harvesting Robot," *J. Biosyst. Eng.*, vol. 48, no. 2, pp. 178–197, Jun. 2023, https://doi.org/10.1007/s42853-023-00184-y.
[50] H. Ge, Z. Ying, Z. Chen, W. Zu, C. Liu, and Y. Jin, "Improved A* Algorithm for Path Planning of Spherical Robot Considering Energy Consumption," *Sensors*, vol. 23, no. 16, p. 7115, Aug. 2023, https://doi.org/10.3390/s23167115.
[51] R. Kabir, Y. Watanobe, M. R. Islam, and K. Naruse, "Enhanced Robot Motion Block of A-Star Algorithm for Robotic Path Planning," *Sensors*, vol. 24, no. 5, Art. no. 5, Jan. 2024, https://doi.org/10.3390/s24051422.

## BIOGRAPHY OF AUTHORS

**Ira Kumalasari** earned her B.Eng. degree from Universitas Negeri Malang, Malang, Indonesia, in 2014. She currently works as a staff member in the Control and Digital Systems Laboratory and is part of the research group in the Control and Robotics Laboratory at Universitas Negeri Malang. Her research interests include control systems, artificial intelligence, machine learning, and robotics. Email: ira.kumalasari.2205348@students.um.ac.id.

**Harits Ar Rosyid** is a lecturer and researcher in the Department of Electrical Engineering and Informatic at Universitas Negeri Malang (UM), Indonesia. He earned his bachelor's degree from Universitas Brawijaya, master's from Institut Teknologi Bandung, and Ph.D. from the University of Manchester. His research interests include game technology, game development, educational and serious games, machine learning, and computer vision. Email: harits.arrosyid@um.ac.id. ORCID: 0000-0002-0014-3533.

**Siti Sendari** earned her B.Eng. from Universitas Brawijaya in 1996, and her M.Eng. from Universitas Gadjah Mada in 2005. She completed her Ph.D. at Waseda University, Japan, in 2013. Currently, she is a lecturer at Universitas Negeri Malang, leading the Dynamic Systems, Control, and Robotics research group. Her research focuses on signal and systems, control, AI, machine learning, and robotics. Email: siti.sendari.ft@um.ac.id. ORCID: 0000-0002-8681-7024.

**Norrima Binti Mokhtar** earned her B.Eng. in Electrical Engineering from Universiti Malaya in 2000. After working in international telecommunications, she received a Panasonic Scholarship and completed her M.Eng. in Japan in 2006. She later earned her Ph.D. in 2012 with a SLAB/SLAI scholarship. Currently, she is a lecturer at Universiti Malaya, with over 50 publications in robotics, automation, and related fields. She has supervised four Master's and seven Ph.D. students and actively reviews papers for international journals and conferences. Email: norrimamokhtar@um.edu.my. ORCID: 0000-0002-2839-1336.

**Samsul Setumin** earned his B.Eng. from the University of Surrey in 2006, M.Eng. from Universiti Teknologi Malaysia in 2009, and Ph.D. in Imaging from Universiti Sains Malaysia in 2019. He has been a lecturer at Universiti Teknologi MARA (UiTM) Pulau Pinang since 2010. His research interests include embedded systems, pattern recognition, computer vision, and image processing. Email: samsuls@uitm.edu.my. ORCID: 0000-0002-9391-4092.