Application of Graph Theory on Dublin Airport Management

Laxman Singh Doliya
ID:20244665
Research Project
MSCDAD JAN22 A

Dec 2022

Abstract

The airport management face operational challenges with the growth in the number of passengers, limited infrastructure, congestion in terminals, and increasing prices. Although significant progress has been made in the field of passenger analytics such as identifying crowds in the airport and predicting the inflow of passengers, but real-time management congestion in airports remains an ongoing problem. The deep learning methodologies combined with graph theory applications can be applied to tackle the problem of managing crowds in the airport. The cognitive abilities of deep learning methodologies identify the formation of crowds and graph theory is used to provide a solution to manage the crowds by converting the airport space into a two-dimensional graph. In this research paper, the research proposes a hybrid solution to manage the crowds using the CSRNet algorithm to identify the density of the crowds and then using the A* path-finding algorithm to provide the optimal path for the passengers to reach from source to destination. A simulation model verifies the effectiveness of the proposed solution. The proposed solution creates an intelligent solution that provides a dynamic path based on the density of the crowd within a given region. This research will enable the authorities of Dublin airport to better manage the crowd and thereby provide outstanding services to the passengers using the airport facilities.

1 Introduction

As the demand for air transport increases, the need for the management of airport processes becomes essential for the well-being of the aviation industry. One of the crucial aspects of airport management is the management of passengers traveling through the airport. The airport is constrained by tangible resources like the space inside the airport, the number of airport personnel, and the number of flights scheduled, and intangible resources like weather conditions, long weekends, and holiday seasons. The huge influx of passengers exerts pressure on management as the resources are limited and therefore the quality of service degrades as the number of passengers increases. The decision-making process in airport management is made swift by the use of real-time data from the airport. The Heathrow's Airport Operation Centre of the Heathrow airport in London deployed real-time analytics using machine learning (Guo et al. 2020) to model passenger flows in an airport. The study supports how the use of predictive algorithms and data flow can help influence better decision-making. The study claims that Group ADP (formerly Aeroports De Paris) contacted the Heathrow's Airport Operation Centre to replicate the solutions at Charles de Gaulle, Orly Airport, and Le Bourget Airport.

As per the report (OxfordEconomics 2018, p. 2), over the next five years, the number of passengers using Dublin Airport is anticipated to climb more fast, rising by an additional 1.7 million to 62.5 million until 2050. The report (OxfordEconomics 2018, p. 4) highlights the need for additional capacity to meet the demands by the year 2050 and a possible need for a new terminal (OxfordEconomics 2018, p. 100). The report however does not address the solutions that can be used to tackle the current situation as described in news article (Elton 2022), that is, using real-time data analytics to manage the crowd seamlessly with resources at hand.

The previous research utilized real-time data-driven analytics to provide a descriptive view of crowd management such as the formation of crowds at certain areas within the airport, the expected number of passengers, and automated reporting. The previous research lacked in providing solutions to the problems in real time. The research aims to provide a solution to crowd managementreal-time the graph theory algorithms to provide the optimal path for passengers based on real-time data and ensure that no adjacent boarding stations have simultaneous take-off or landing scheduled for those boarding stations. A convolution neural network using CSRNet (Congested Scene Recognition Network) is used to identify the density of the crowds, the A* path-finding algorithm is used for getting the optimal path from a giver source to the desired destination, and the graph coloring algorithm to assign different colors to the boarding gates so that no adjacent gates have passengers coming at the same time.

The results of the research are affected by the number of people arriving at

the airport, the quality of images provided by the cameras of the airport, and the effectiveness of the simulation model to replicate the actual scenario.
Research Question Can a hybrid data mining model combining crowd density detection and graph theory algorithms be used to find the optimal path in crowd-congested areas inside the airport?
The objectives of the research is to:

- Design a CSRNet model to identify the group of crowds in the airport.
- Design an OpenCV method to convert the density map images to grid-graph representation.
- Implement the A* path-finding algorithm to find the optimal path.
- Implement a graph coloring algorithm so that no adjacent boarding gates have passengers arriving at the same time.

The research has a limitation where it is not able to provide an optimal path in scenarios where all the source and destination paths are choked by the formation of crowds and where the manual intervention of airport management authorities is required.
The research does not have access to the actual data of the Dublin airport therefore the number of passengers is created using statistical distribution functions based on monthly passenger data available on the website of the Dublin airport. Similarly, for the image data, the research has used the data available on the internet for the congested place. The research aims to provide a near real-life scenario of the airport.
The major contribution of the research is to propose an implementation of a hybrid solution by using crowd image data with CSRNet and the A* path-finding algorithm to provide the optimal path to manage the crowds within the airport.
The research paper is divided into sections and each section serves as an aid to understand the whole research. The Literature Review section discusses the previous research done in the particular area. The Research Methodology discusses in detail CSRNet and A* path-finding algorithm and the rationale behind choosing them in the research. The Design and Implementation specification deals with the infrastructure required to replicate the research. The Evaluations section discusses the use of a simulation model in verifying the advantage of using the proposed hybrid solution. The results and advantages are summarized in the Conclusion section.

2 Literature Review

When many people gather in one area, there are considerable vulnerabilities that, if not adequately managed, could have significant effects. Prior research (Still et al. 2020) discussed the process of crowd science and describe a methodology for crowd safety issue(s) in congested environments. The researchers divided the research into three sections namely, crowd modeling, crowd counting, and crowd monitoring and management. This research used the Canary Wharf Project as a case study where the research proposed the circulation movement of people in the event. In this research, simulation models were used to mimic the crowd scenarios. Visual-based crowd-counting models were used to obtain the demographics of the crowd. Therefore this research provides a solid foundation for addressing the management of crowds.
The research (Widera & Hellingrath 2020) discussed the present-day trends and upcoming challenges. The research highlights two major issues, that are, different components of the project are independent but have different goals and the solutions are guided by protocols that are pre-existing and the structures for the solution cannot be prepared immediately. These two issues are major issues that are common to almost every problem that requires a complex solution. The use of real-time data helps in providing solutions for businesses as they happen.

2.1 Crowd Density Detection

The research (Li et al. 2018, p . 1092-1093) discusses the different categories of crowd estimation techniques namely, detection-based, regression-based, and density-based approaches. While the detection-based approach fails for highly congested areas and the regression-based approach fails due to inaccurate predictions arising due to different counts of people for the same geometric area, the detection-based approach overcomes the issue by learning the relationship between local features and object density maps. In our research, a density-based approach is used as the goal of the research is to obtain the optimal path by navigating through the crowd. The density maps give us the spread of crowds within the airport. The research proposed a state-of-the-art deep convolutional neural network called CSRNet (Congested Scene Recognition). Due to the ability of CSRNet to output density maps along with the estimate of the number of people in crowds, CSRNet is used in our research.
The images and/or videos collected by cameras vary in scale. The research (Kuchhold et al. 2018) provided a solution that uses FAST (Features from Accelerated Segment Test) which is a corner detection technique to overcome the scale variation issue. The use of FAST features is a smart idea as it overcomes the variation due to the angle of the cameras. The FAST required high-resolution images for

training and since it is a corner detection technique, the objects require a stark difference in pixel intensities. The airport areas have varied lighting in different areas and therefore in areas that are not well illuminated, the pixels do not vary drastically.
The research (Zhao et al. 2019) discussed problems like scale variation, clutter background, and global count for crowd estimation models. The size of humans near to camera is larger as compared to the ones at the background. Thus, there is a difference in densities in images which cannot be addressed using a single mathematical model. The idea to use the image in segments is a divide-and-conquer approach that allows CNN to learn features in a better way. This is an advantage for our CNN model as it makes the model aware of the intra-variation within the image of the crowd.
The refinement of density maps is discussed in the research (Wan & Chan 2019). The researcher used three techniques to refine the density maps namely, gaussian blurring, self-attention, and fusion. The advantage is that the density maps could be directly generated without having to use intermediate ground truth data. This reduces the training period for the crowd estimation models.
Another variation of CSRNet, called Soft-CSR is discussed in the research (Bakour et al. 2021). The frontend CNN has a VGG16 layer that is a pre-trained model. But the backend model has 5 layers of convolution. The little variation in the backend layer leads to the creation of a model that has less time complexity.
The research (Papaioannidis et al. 2021) discussed the use of semantic segmentation and image-to-image translation for crowd detection by UAVs. Semantic segmentation is used to learn about spatial information which is essential for continuous video feeds. The advantage of using this approach is that it can be utilized in areas of airports that have large spaces and the distance from the camera to passengers is quite large.

2.2 Path Finding: Graph Theory

The research (Goldberg & Harrelson 2003) discussed the A* path-finding algorithm and compared it to the classic Dijkstra algorithm. The research also includes landmarks in the graph that are nodes lying in the path of the source node to the destination node. Since the A* algorithm takes into account the cost to select the optimal path, the research selects the A* algorithm to find out the best path in our research.
The graph theory can be used for finding the paths for the internet of vehicles, as discussed in (gan Zhang et al. 2019). The use of dynamic links is adequate in this research and is replicated in our research where the different paths between source and destination depend on the formation of crowds within the airport.
Another use of the path-finding algorithm for robots using the graph theory is dis-

cussed in (Cheng et al. 2019). The research deployed the Dijkstra algorithm with backtracking to feed the path to the robot. The graph is generated using a grid approach where the obstacles are non-path blocks within the grid. Unlike conventional graphs, which have directed nodes between nodes, the grid act as a link. Additionally, the cost of movement is calculated by multiplying the unit cost of moving from one grid block to another adjacent block with the number of blocks traveled.
The research (Song et al. 2019) discussed the advantage of using smoothed A* algorithm. The (Song et al. 2019, p . 11) highlights the disadvantages of the A* algorithm such as a huge number of sharp turns in the proposed path. Path smoothing techniques namely, line-of-sight path smoother, waypoint refining path smoother, and interpolation path smoother were used in the research. The advantage of using smoothing is that it provides a path that is easier to travel and collisions can be avoided on sharp turns.
The research (Zhong et al. 2020) discussed another approach to provide a collision-free path by using an adaptive window approach with the A* algorithm. The advantage of an adaptive window is that it searches for all possible angles at which the next step is to be taken. An angle that provides a smooth path is chosen. The angle selected also provides the change in velocity required.

2.3 Graph Coloring Algorithm

The research (Giorgetta et al. 2006) explained the use of graph coloring with backtracking for Filed Programmable Gate Array systems that hot-swap application-specific modules without distracting other operations. Backtracking minimizes the number of colors that are used to color by revisiting the previous color assignment and assigning the color to the current node if a swap can help to choose an already used color instead of new color. The backtracking although reduces the number of colors, but backtracking can take a long time especially when the graph has a depth.
The use of graph coloring for flight gate assignment for an entire terminal has been discussed in the research (Li et al. 2019). The use of flight time conflicts and flight assignment difficulty coefficient provides a realistic scenario. The research used constraints like a safety interval of 45 minutes between two successive flights and parking of smaller flights on bigger boarding gates while the converse not being true.
The research (Das et al. 2020) used deep learning algorithms to approximate the number of colors for graph coloring problems. An LSTM model is used to provide a global minimum for the number of colors to color the graph nodes. The constraints were not considered in the research while applying the LSTM. Hence, the number of colors provided by the LSTM is sub-optimal.

Another research (Stollenwerk et al. 2021), approached the flight gate allocation problem using the graph coloring algorithm with quadratic binary function with constraints that follows quantum annealing. The strong point in the research is the use of partially controlled color exchange that swaps colors in the graph which is controlled by the nodes adjacent to the current node. The research provides a minimum number of colors but fails in special cases and thus cannot be used in all scenarios.

2.4 Simulation

The evacuation of a school using agent-based simulation is discussed in the research (Poulos et al. 2018). The research effectively used personal attributes like differences in speeds of 4-6 years old students and students above 6 years in age, using Weibull distribution for the arrival of students for evacuation, and reducing the speed on stairs by 50%. The use of attributes created a realistic simulation which is further evident in the research when the simulation and actual drill had near-similar outputs. Thus simulation help as an alternative to implementing the solution before actually rolling it out.
In their research for the mobile ad-hoc networks under failure processes, the simulation is used to study the average shortest path length in (Liu et al. 2019, p . 21352-21356). The use of graphs generated after running the simulations model gives us a contrast under a different set of conditions. Thus, the advantage of simulations is different conditions are studied giving us an estimate of the efficiency of the solution.
The research (Hao et al. 2020) used agents to the efficiency of a navigation model that provided commands using visuals and natural language. The use of agents allows the researcher to fine-tune the model for a specific layer in the model. It also allows the extraction of features from a pre-trained model and adds a layer to the model. The advantage of using agent-based simulation is that it allows us a scope to make or roll out a change and study its impact before the actual implementation.

3 Research Methodology: Design and Implementation

The research uses different components that interact with each other in a sequential manner to provide the best solution. The steps used in the research are:

1. Obtain a map of Dublin airport to find the region of interest for the problem.

2. The crowd density estimation model is created using a state-of-the-art convolutional neural network: CSRNet. The model identifies the blobs or densities of the crowd within the region of interest.

3. A 2D grid graph combining the map of the region of interest in the airport and hotspots of crowd density in that region is created using OpenCV. The goal is to find the best path using grids that do not have any obstacles.

4. The A* algorithm is applied to the grid graph obtained in the above step to finding the optimal path in the given region.

5. The map of boarding gates is used and converted into an adjacency matrix. The graph coloring algorithm is applied to assign different colors to the boarding gates so that no boarding gates adjacent to each other have the same color. Different color ensures that no adjacent boarding gates are used as boarding gates simultaneously and allowing free movement of passengers and lowering the risk of formation of crowds.

Figure 1 shows the flow of the entire research methodology. The research follows Knowledge Discovery in Databases(KDD) methodology.

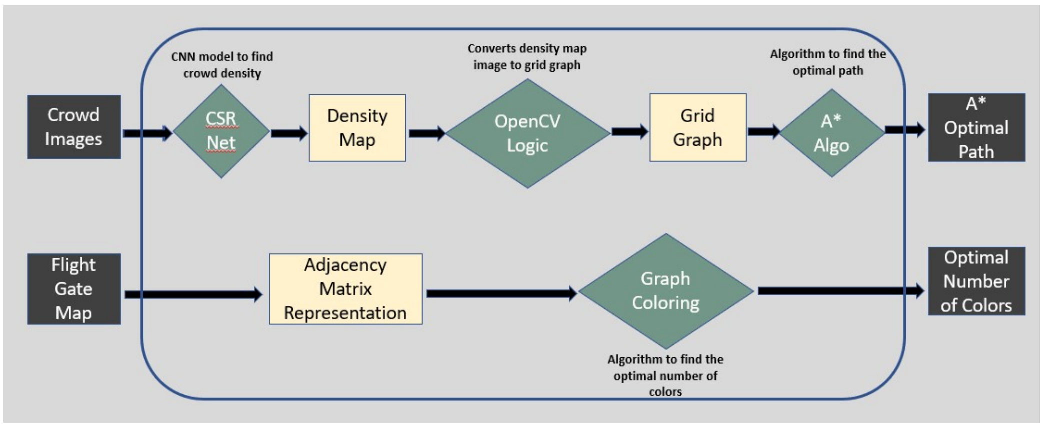

Crowd Images
CNN model to find crowd density
CSR Net
Density Map
Converts density map image to grid graph
OpenCV Logic
Grid Graph
Algorithm to find the optimal path
A* Algo
A* Optimal Path
Flight Gate Map
Adjacency Matrix Representation
Graph Coloring
Algorithm to find the optimal number of colors
Optimal Number of Colors

Figure 1:
The Research Methodology Flow Chart

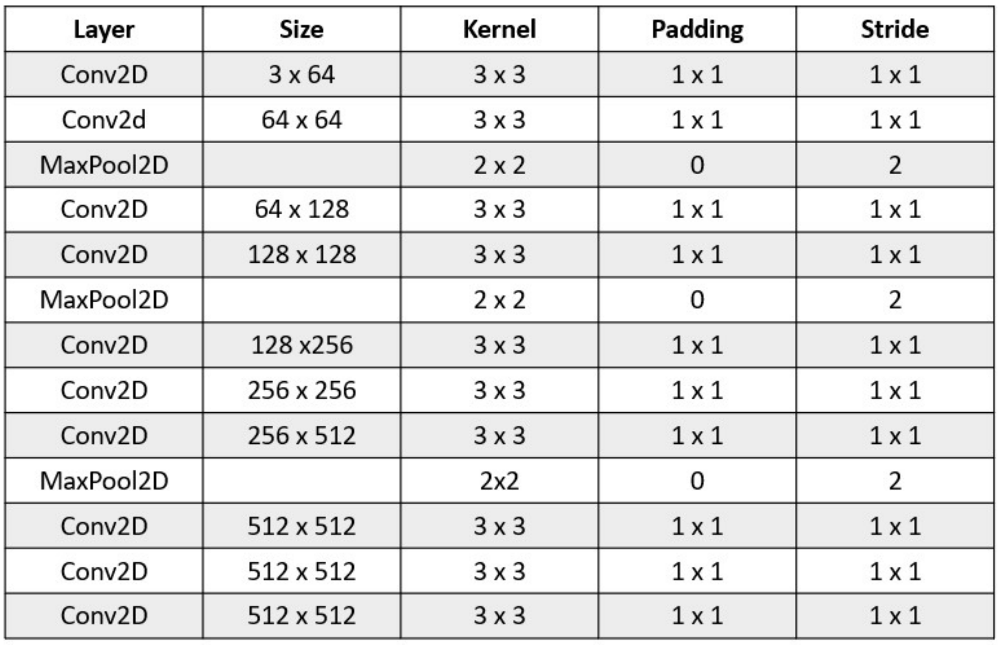
3.1 CSRNet: Crowd Density Estimation

In our research, a density-based approach known as CSRNet is used to identify the number of people in the crowd and generate their density maps.
Dataset: The Shanghai crowd dataset which is a public dataset [1] is used to train

[1]Dataset Link: `https://www.kaggle.com/datasets/tthien/shanghaitech`

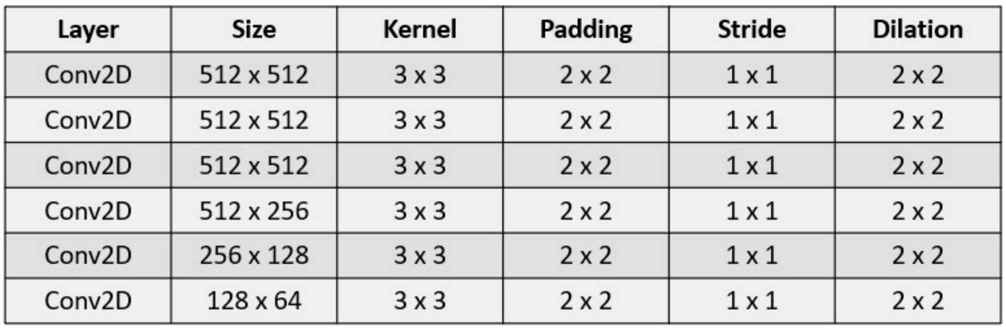
the CSRNet model and is obtained from Kaggle (ShanghaiTech Dataset Link). The dataset appeared first in the research (Zhang et al. 2016) and contains the actual images of the crowds with the faces of people blurred. The dataset is divided into two parts. The dataset consists of actual images and ground truth data. The ground truth(.mat file format) contains information like the (x,y) coordinates of the head of people in the crowd as on a 2D cartesian plane and the actual number of people in the corresponding image.
Data Augmentation: Random horizontal flip is used to flip the image within a probability range. The research used *probability range of 0.5 for the random flip*. The paired crop is used to crop images at specific locations. The research used a *factor of 16 to get random cropped areas of size 1/16th* of the image.
Design: The model uses a pre-trained VGG16 model for the front end. The back end is used for back propagation. In its back-propagation, model tries to update the filter values, adjusting them to be more effective at fitting the output and aiding in learning. In order to train the model, the density maps matrix of the image are generated. The model learns the relationship between the density maps matrix and the actual number of people to approximate the number of people in any given image of the crowd. The basic concept behind the suggested design is to use a deeper CNN to capture high-level features with broader receptive fields and produce superior-quality density maps without drastically increasing network complexity.
Implementation: The first step is to density map for each image in training set. The ground-truth images contain the coordinates of the head of the people in the crowd. A gaussian-filter is used on the coordinates of the head of people. The gaussian filter computes a weighted average of the neighbouring pixels on the basis of the normal distribution. The research uses a sigma value of 15 that is used as the size of the kernel. The gaussian filter returns a numpy matrix which is used to create the density map data.
Model creation and parameters: The research uses a pre-trained VGG16 model along with transfer learning to train the model with our crowd dataset. To create a CSRNet model, additional 10 layers are added to the VGG16 model to create the front end.
The kernel in convolutional layers is used to scan the pixels to convert data into a smaller or larger format. The padding is added to convolutional layers to assist the kernel to move more freely over the pixels especially the pixels at the corners of the image. The stride defines the amount of movement of the kernel.
The dilation is used in the back end of the crowd density estimation model. The dilation is used because it does not cause a loss in the resolution of the output image. This is required as the images captured by cameras have differences in the size of the heads. The people near have large heads as compared to the ones at the back, thus loss in resolution causes loss of information as well. Dilation has

an advantage over the max-pooling layer which chooses the pixel with the highest intensity and thus loses valuable information in the process.
The architecture of the front end is shown in figure 2. The back end of the CSRNet consists of 6 layers as shown in the figure 3.



Layer	Size	Kernel	Padding	Stride
Conv2D	3 x 64	3 x 3	1 x 1	1 x 1
Conv2d	64 x 64	3 x 3	1 x 1	1 x 1
MaxPool2D		2 x 2	0	2
Conv2D	64 x 128	3 x 3	1 x 1	1 x 1
Conv2D	128 x 128	3 x 3	1 x 1	1 x 1
MaxPool2D		2 x 2	0	2
Conv2D	128 x256	3 x 3	1 x 1	1 x 1
Conv2D	256 x 256	3 x 3	1 x 1	1 x 1
Conv2D	256 x 512	3 x 3	1 x 1	1 x 1
MaxPool2D		2x2	0	2
Conv2D	512 x 512	3 x 3	1 x 1	1 x 1
Conv2D	512 x 512	3 x 3	1 x 1	1 x 1
Conv2D	512 x 512	3 x 3	1 x 1	1 x 1

Figure 2:
The front end of CSRNet.



Layer	Size	Kernel	Padding	Stride	Dilation
Conv2D	512 x 512	3 x 3	2 x 2	1 x 1	2 x 2
Conv2D	512 x 512	3 x 3	2 x 2	1 x 1	2 x 2
Conv2D	512 x 512	3 x 3	2 x 2	1 x 1	2 x 2
Conv2D	512 x 256	3 x 3	2 x 2	1 x 1	2 x 2
Conv2D	256 x 128	3 x 3	2 x 2	1 x 1	2 x 2
Conv2D	128 x 64	3 x 3	2 x 2	1 x 1	2 x 2

Figure 3:
The back end of CSRNet.

The model is trained *root mean square loss function* that adjusts the weights of the hidden layers. *The research trained our model for 30 epochs and stored the configuration of the epoch with the lowest Mean Absolute Error*. The configuration of the best epoch is then loaded into the CSRNet at a later time to make the predictions.
Figure 4 shows the actual image and density map returned by our CSRNet model.

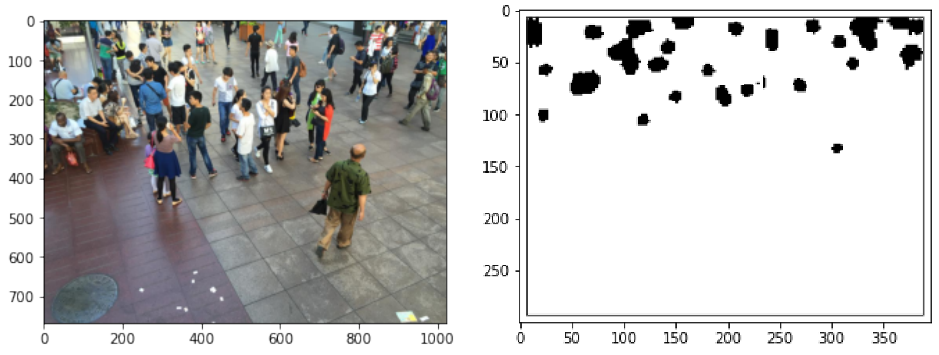


Figure 4:
Original Image vs Density Map returned by CSRNet model

3.2 OpenCV to convert density maps to grid graph

The A* path-finding algorithm requires the data in a list of list format with each element in the list specifying if the position is a path or an obstacle. Therefore, the researcher converts the image into a grid-type format, where each square in the grid represents if the square is a path or an obstacle. OpenCV is a python library that is used for image processing and analysis. The density map returned by our CSRNet model is an image. The researcher creates the graph using a grid so that the path provided by the research allows for free movement, similar to how humans move in reality.
Dataset: The density map images returned by CSRNet.
Design: The design phase starts by reading the image. The crowd density is then highlighted using image thresholding. Image thresholding is a technique to partition image into foreground and background. The foreground in our research is the crowd densities and background is the rest of the image. A threshold value is used to partition the image. If the value of pixel is image is above the threshold value, it is given higher intensity so that the pixel has a black color. If it is below the pixel value it is given white color. The next step is to partition images into smaller regions. The research computes the average color of the region. If the average is

equal to zero, consider it an obstacle. The output is saved to a text file so that it can be referenced later.
Implementation: The image is read using OpenCV. For image threshold, a threshold value of 200 is used, so that even pixels with light grey color are treated as an obstacle. The image is partitioned into regions using a nested loop. The NumPy library's average function is used to calculate the average value of each region. If the average is 0, it is considered a crowd or an obstacle, and write the data to a text file with '$' value, and with '-' otherwise. The text file is then read line by line into a list of lists, with each element list representing a square in the grid. Plot the grid and visualize the grid graph representation.
Figure 5 shows the grid graph obtained. The black squares are the path and the red squares are the obstacles. For reference, the yellow square in the grid is the starting position and the white square is the destination.

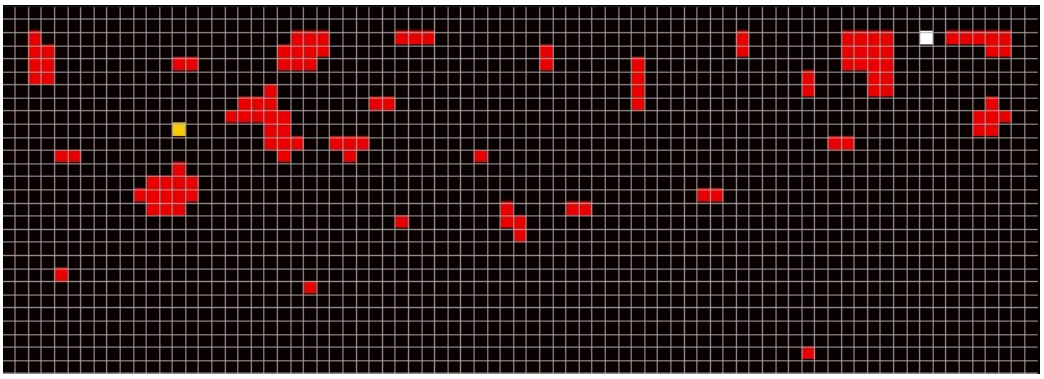


Figure 5:
Gird Map with Source-Destination

3.3 A* Path-Finding Algorithm

Dataset: The text file containing path and obstacle returned by OpenCV logic.
Design: The choice of movement to a neighboring square in a particular direction is based on the cost known as F-cost. The F-cost is the summation of two costs called G-cost and H-cost. The G-cost is the exact distance from the source square in the grid to the current square. The H-cost is a heuristic cost, which is the exact opposite of the G-cost. The H-cost is the distance of the current square in the grid from the destination square. If the algorithm, finds two or more squares with the same F-cost, the algorithm visits each one of them and looks for the updated F-costs of its neighboring squares. The method maintains two lists to keep track

of the visited squares and unvisited squares. The squares are moved in and out of the list if the algorithm chooses another square for the movement that provides a lower F-cost.
Implementation: Initially a class Node which is used to store the coordinates of each square on the grid and provide a move cost is created. A *move cost of 1* is used for a move from one square to an adjacent square. The possible movement links are calculated. A link is possible if the square we are moving to is not an obstacle. A *four direction move* is used. The eight directions move is not used to avoid any collision on the edges of the crowd densities. Initially G-cost is set to zero. With each possible move the G-cost is updated by 1. The research has used the *Manhattan distance* to calculate the F-cost. The algorithm chooses the next square on the basis of the lowest F-cost until it reaches the destination. Figure 6 shows the result obtained to reach point A to point B.

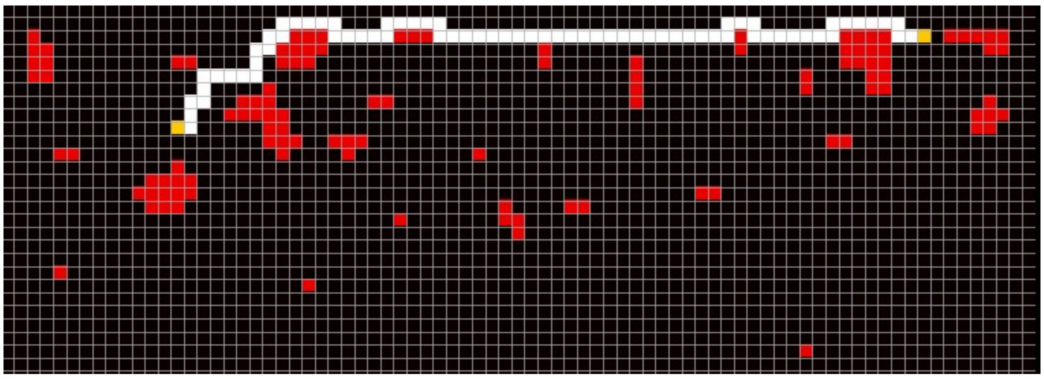
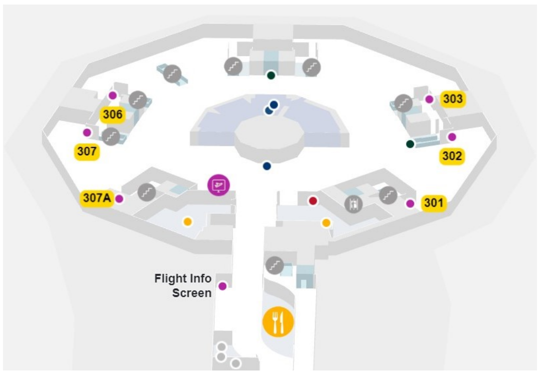
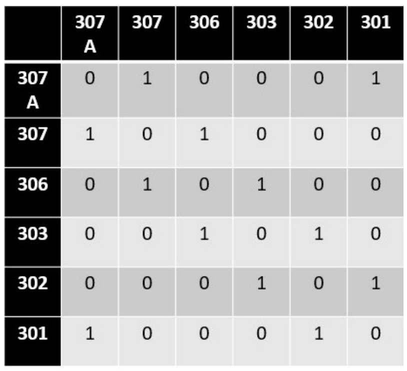


Figure 6:
A* Path to follow

3.4 Flight Gates allocation using Graph Coloring Algorithm

Dataset: The flight gate allocation requires the map of the boarding gates in the form of an adjacency matrix. A square matrix is created with a dimension equal to the total number of flight gates, with values in 0 or 1. For a flight gate, for example, gate A, if gate B and gate D are adjacent to it, put a 1 in their corresponding location, and the rest are 0 as shown in Figure 7.
Design: The degree for each boarding gate is calculated by finding the number of adjacent gates to it. Sort the boarding gates in descending order. An iterative process assigns a color to each gate. The gate in the sorted list is assigned the first color. A nested loop with an outer loop running for the list of degree-sorted gates and an inner loop running for the length of the row of the current gate in the process. For the gate that is now being selected by the outer loop, the loop gives

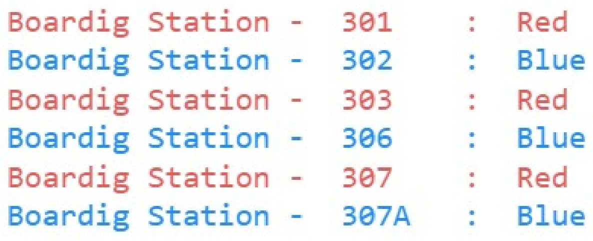




	307 A	307	306	303	302	301
307 A	0	1	0	0	0	1
307	1	0	1	0	0	0
306	0	1	0	1	0	0
303	0	0	1	0	1	0
302	0	0	0	1	0	1
301	1	0	0	0	1	0

Figure 7:
Terminal 1 and corresponding adjacency matrix

the neighboring gate a color. Each iteration of the loop ensures the adjacent node has been assigned a different color.
Implementation: The research used boarding gates of T1 of Dublin airport. The adjacency matrix is created manually. Take six different colors as the terminal has six flight gates. The degree for each gate is calculated and sorted using *the selection sort*. A nested loop assigns the color with the outer loop running for each row in the matrix and the inner loop for the length of each row. Figure 8 shows the color assigned to each gate.



```
Boardig Station -  301    :  Red
Boardig Station -  302    :  Blue
Boardig Station -  303    :  Red
Boardig Station -  306    :  Blue
Boardig Station -  307    :  Red
Boardig Station -  307A   :  Blue
```

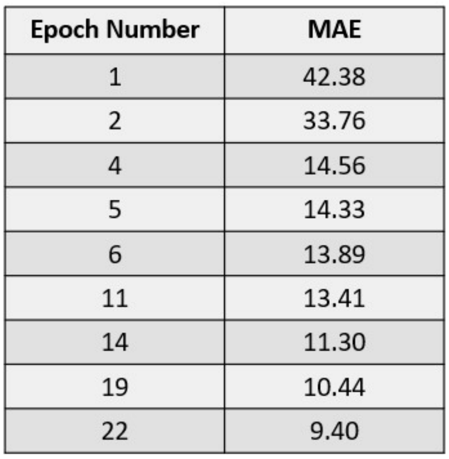
Figure 8:
Flight Gate Color

4 Evaluation

The efficiency of the proposed model is checked using the simulation model. The research also evaluates the crowd density model used in the research. The A* algorithm achieves both optimality and completeness. Optimality refers to the ability of the algorithm to find the best possible solution. Completeness refers to the ability of the algorithm to find the solution if there exists one. Therefore, the research cannot evaluate the algorithm on any metric. Similarly, the graph coloring algorithm is an NP-complete problem. NP-problem is a problem for which the solution exists but it does not have a definite polynomial time. The research Garey & Johnson (1976) discusses the complexity of graph coloring algorithms. However, due to the simpler design of flight gates in the Dublin airport, the algorithm is able to find the minimum number of colors easily.

4.1 CSRNet: Evaluation using Mean Absolute Error

The mean absolute error is the average of all the errors returned during the training phase of the CSRNet model. To evaluate the model, the author only uses forward propagation layers in the CSRNet. This is because backward propagation layers are used to adjust the weights of the hidden layer. The specific layers like dropout and batchnorm layers are turned off. The *model.eval()* is used to switch off these layers. The *torch.no_grad()* is used so the model only updates the weights in the training phase and forgets the gradients as they are not required during the evaluation of each epoch. For each image used for testing, calculate the error by subtracting the density value returned by the model from the actual ground truth density value. This sum value is divided by the total number of test images. Figure 9 shows the execution of epochs until epoch 22 for which our model had the least mean absolute error.



Epoch Number	MAE
1	42.38
2	33.76
4	14.56
5	14.33
6	13.89
11	13.41
14	11.30
19	10.44
22	9.40

Figure 9:
MAE of the best epoch 22

4.2 Simulation: Evaluation of Research

The path provided by the A* algorithm is verified as optimal by comparing it to the path provided by another greedy path-finding algorithm. An agent-based simulation technique is used to compare the paths provided by the A* algorithm and the greedy path-finding algorithm.

Why Greedy Path-Finding Algorithm for human behavior: The greedy algorithm divides a complex problem into smaller sub-problems. The solution to sub-problems is found more easily as compared to a large complex problem. These sub-solutions then are combined for the solution to the complex problem. Instinctively humans behave in a similar manner as discussed in the research (Prasanth et al. 2020, p . 2831). While navigating through the crowded space, the passengers try to move into spaces that are more convenient and navigate their path as they move along into these convenient spaces.

Why Agent-Based Simulation: The agent-based simulation is used to model an individual who can imitate the behavior using the properties and behavior assigned to it. Since both the A* algorithm and the greedy algorithm model the path taking into account the cost of moving, and the crowd formation inside the given region, the agent-based simulation are best fit to evaluate the research. Furthermore, the agent-based simulation best capture the dynamic behavior of the pedestrians as discussed in the research Rozo et al. (2019).

Implementation: The research creates the greedy approach by taking into account only the G-cost. The G-cost is similar to the one used in the A* star algorithm. Since the G-cost is the distance from the source to the adjacent square in the grid, the passenger moves into the square that is least far and is not an obstacle. The algorithm then finds the next square with the least distance value. A class Node is created to create an object that holds attributes of the square of the grid like its coordinates and the parent that is used to hold the square from proceeding the object in the path. A method is written to check the path provided by the greedy algorithm so that it does not try to find a path through the crowds in the region. If so is the case, the path is truncated and the path terminates on the crowd the path reaches first. This method highlights the case where the greedy algorithm does not find a path, but rather contributes to the growth of crowd formation. The researcher created a method that calculates the number of moves taken by the paths provided by the A* algorithm and the greedy algorithm.

Figure 6 showed us the path returned by the A* algorithm.

Figure 10 shows the path returned by the greedy algorithm which is longer than the one returned by the A* algorithm. Figure 11 shows the case where the path returned by the greedy algorithm does not reach the destination but adds to the crowd.

Figure 12 compares the A* and greedy path-finding algorithms.

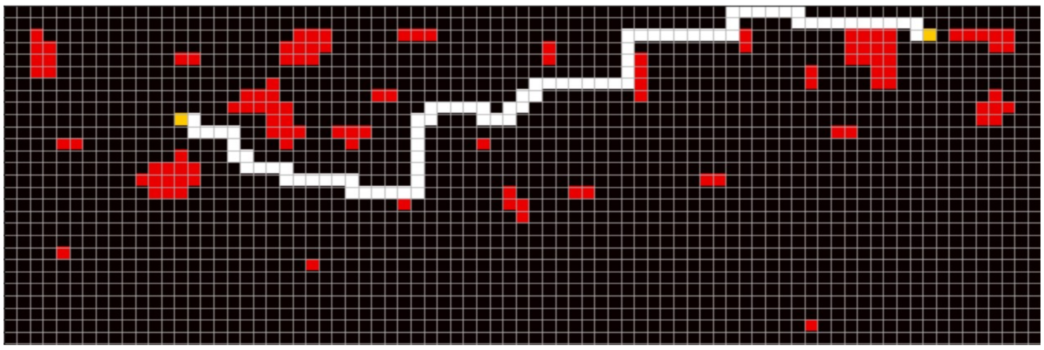


Figure 10:
The path returned by the greedy algorithm

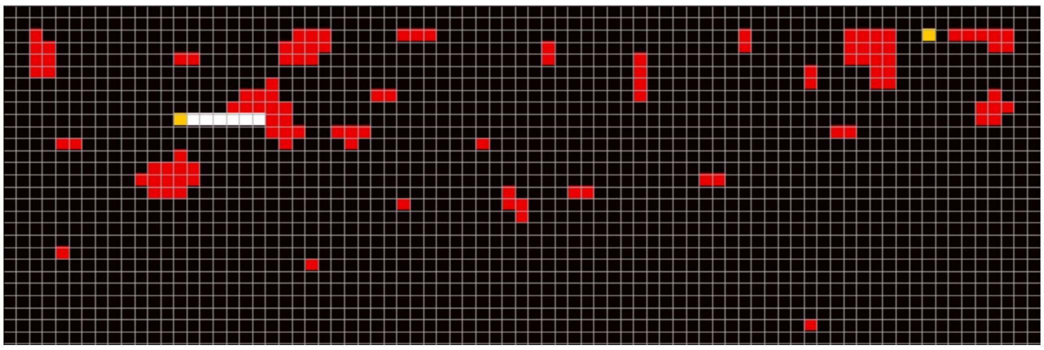
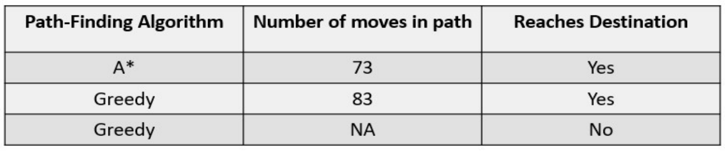


Figure 11:
The path returned by the greedy algorithm that reaches a crowd



Path-Finding Algorithm	Number of moves in path	Reaches Destination
A*	73	Yes
Greedy	83	Yes
Greedy	NA	No

Figure 12:
Comparison of path-finding algorithms

Limitation: The research does not evaluate the crowd-counting model that is able to detect other objects like flight check-in desks, inquiry desks, luggage, and escalators. Another aspect lacking in the research is checking the width of pathways and providing a path that penalizes the capacity of pathways. The analysis of images from the airport is not used as the actual data is not present due to security reasons. The simulation model lacks the capability to handle the dynamic changes in the environment like updating the grid graph with the influx of more passengers.

5 Experiment

The experiments are conducted to establish the correctness of the path-finding problem. The researcher conducts three experiments with different crowd scenarios and a different set of sources and destinations for each scenario. The source in the grid graph is the yellow square and the destination is the white square. However, while plotting the path both the source and the destination are yellow squares, and the path is plotted using white squares. The grid has 60 rows and 79 columns of data.

5.1 Experiment 1

Figure 13 shows from left to right, the original image used for the scenario, the corresponding density map, and the grid graph representation. The start location for the path is (17,63) and the destination to reach is (7,61). Figure 14 shows the path returned from the A* algorithm, simulation1, and simulation2 of the greedy path-finding algorithm.

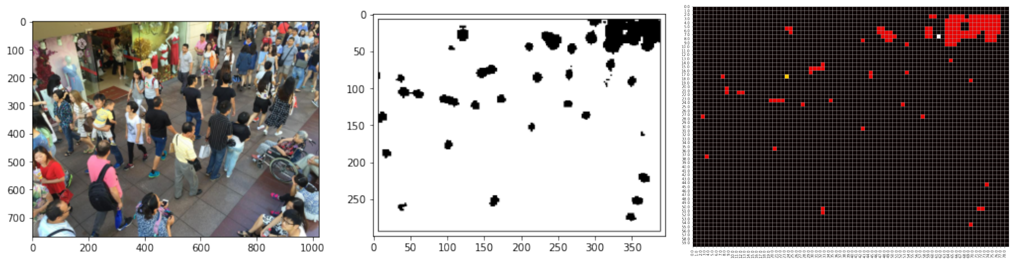


Figure 13:
Original Image(left), Density Map(middle), and Grid Graph Representation(right)

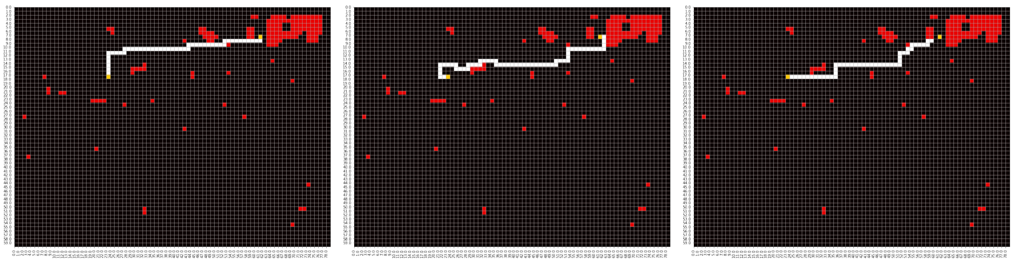


Figure 14:
A* Path(left), Simulation 1 path(middle), and Simulation2 path(right)

5.2 Experiment 2

Figure 15 shows from left to right, the original image used for the scenario, the corresponding density map, and the grid graph representation. The start location for the path is (5,69) and the destination to reach is (33,23). Figure 16 shows the path returned from the A* algorithm, simulation1, and simulation2 of the greedy path-finding algorithm.

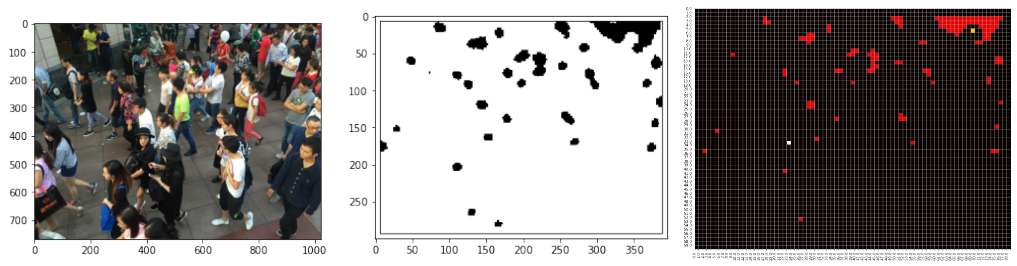


Figure 15:
Original Image(left) and Density Map(middle) and Grid Graph Representation(right)

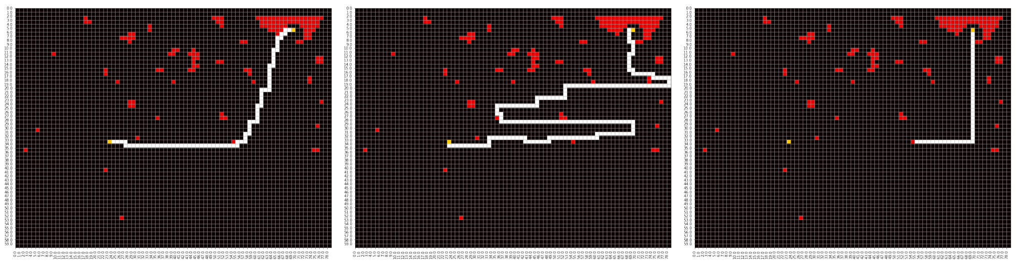


Figure 16:
A* Path(left), Simulation 1 path(middle), and Simulation2 path(right)

5.3 Experiment 3

This is the final experiment conducted for the comparison of the A* and the greedy path-finding algorithm. This experiment tries to find the path for the start location to the right and the destination to the left. Figure 17 shows from left to right, the original image used for the scenario, the corresponding density map, and the grid graph representation. The start location for the path is (20,45) and the destination to reach is (0,15). Figure 18 shows the path returned from the A* algorithm, simulation1, and simulation2 of the greedy path-finding algorithm.

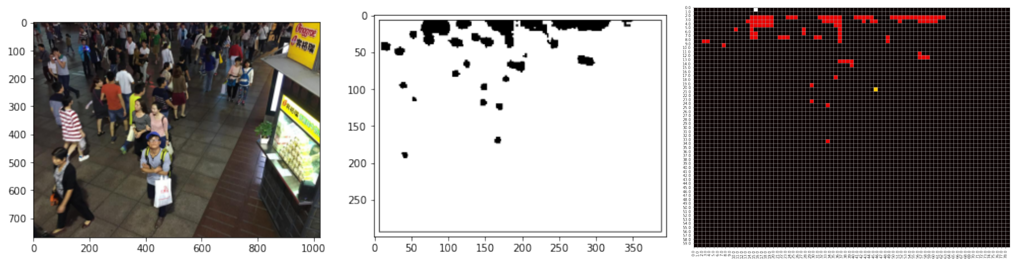


Figure 17:
Original Image(left) and Density Map(middle) and Grid Graph Representation(right)

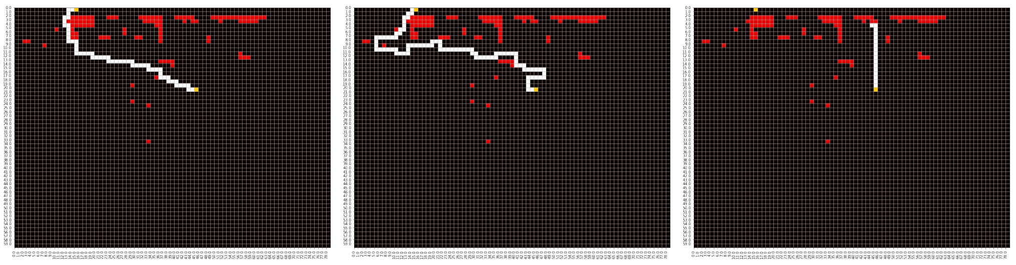


Figure 18:
A* Path(left), Simulation 1 path(middle), and Simulation2 path(right)

Figure 19 summarizes the result of all the above-conducted experiments. The index of the image is the index in the folder where test images are kept. The A* algorithm outperforms the greedy algorithm in all three experiments. The path provided by the A* algorithm takes less number of moves as compared to the greedy path. Further, the greedy path contributes to the formation of the crowd.

	Index of Image in test set	Source row<space>col	Destination row<space>col	A* Path	Simulation 1 of Greedy Algorithm	Simulation 1 of Greedy Algorithm
Experiment 1	20	17 23	7 61	53	87	Reaches Crowd
Experiment 2	70	5 69	33 23	75	225	Reaches Crowd
Experiment 3	200	20 45	0 15	57	87	Reaches Crowd

Figure 19:
Summary of the experiments 1, 2, and 3

6 Conclusions and Discussion

The research created a hybrid model using the CSRNet convolutional neural network and A* path-finding algorithm. The research provides the density of crowds in different regions which is crucial in planning the management of the airport space in real-time. The research also provides a method to convert an image to a grid-graph representation. The grid representation makes it easy to implement graph algorithms on the images. The A* algorithm finds the optimal path connecting the source to the destination in minimal steps. For a particular time, the boarding gates that have been given the same color by the graph coloring algorithm can be scheduled to operate simultaneously, thus avoiding crowd concentration during flight take-off and landing. The path provided by the A* algorithm can also be used for robotic and automated guidance systems. The research provides a methodology that implements a model where the output of one model acts as an input to another model sequentially. This allows us to provide solutions approximating artificial intelligence systems.
Due to privacy and security issues, the research however does not collect the crowd data of the Dublin airport that would have helped in understanding the dynamics inside the airport accurately. The images of the airport would have helped the actual formation of the crowd inside the airport. The grid graph does not account the obstacles like desks, luggage, escalators, and stairs. The schedule of airport flights can be taken to put constraints on the graph-coloring algorithm to suggest gate allocation more accurately. The simulation model lacks the capacity to capture the influx of passengers and update the graph at regular time intervals.

7 Future Works

The research provides a solution that can be used in applications that require efficient path and resource planning like evacuation from war zones, stadiums, concerts, and artificial-intelligence-guided systems such as self-moving cars, drones, and robots. The virtual-reality-based systems can use path-finding to simulate exploration and industrial tasks before actually investing in them. The methods to incorporate the obstacles inside the given region with crowd densities could be developed to provide a more accurate grid graph for path-finding algorithms.

Acknowledgement

I would like to thank my parents, my professors, and my friends for providing support and guidance for my research.

References

Bakour, I., Bouchali, H. N., Allali, S. & Lacheheb, H. (2021), Soft-csrnet: Real-time dilated convolutional neural networks for crowd counting with drones, Institute of Electrical and Electronics Engineers Inc., pp. 28–33.

Cheng, K. P., Mohan, R. E., Nhan, N. H. K. & Le, A. V. (2019), 'Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots', *IEEE Access* **7**, 94642–94657.

Das, D., Ahmad, S. A. & Kumar, V. (2020), Deep learning-based approximate graph-coloring algorithm for register allocation, IEEE, pp. 23–32.
URL: *https://ieeexplore.ieee.org/document/9307022/*

Elton, C. (2022), "hellish queue' stretches a mile out of dublin airport after thousands miss weekend flights'.
URL: *https://www.euronews.com/travel/2022/05/31/hellish-queues-outside-dublin-airport-in-the-early-hours-as-passengers-try-not-to-miss-fli*

gan Zhang, D., meng Tang, Y., ya Cui, Y., xin Gao, J., huan Liu, X. & Zhang, T. (2019), 'Novel reliable routing method for engineering of internet of vehicles based on graph theory', *Engineering Computations (Swansea, Wales)* **36**, 226–247.

Garey, M. R. & Johnson, D. S. (1976), 'The complexity of near-optimal graph coloring', *Journal of the ACM (JACM)* **23**(1), 43–49.

Giorgetta, M., Santambrogio, M., Sciuto, D. & Spoletini, P. (2006), A graph-coloring approach to the allocation and tasks scheduling for reconfigurable architectures, pp. 24–29.

Goldberg, A. V. & Harrelson, C. (2003), 'Computing the shortest path: A * search meets graph theory'.
URL: *http://www.research.microsoft.comhttp://www.avglab.com/andrew/index.html.*

Guo, X., Grushka-Cockayne, Y. & Reyck, B. D. (2020), 'London heathrow airport uses real-time analytics for improving operations', *Journal on Applied Analytics* .

Hao, W., Li, C., Li, X., Carin, L. & Gao, J. (2020), 'Towards learning a generic agent for vision-and-language navigation via pre-training'.
URL: *https://github.com/weituo12321/PREVALENT*

Kuchhold, M., Simon, M., Eiselein, V. & Sikora, T. (2018), Scale-adaptive real-time crowd detection and counting for drone images, IEEE Computer Society, pp. 943–947.

Li, H., Ding, X., Lin, J. & Zhou, J. (2019), ‘Study on coloring method of airport flight-gate allocation problem’, *Journal of Mathematics in Industry* **9**.

Li, Y., Zhang, X. & Chen, D. (2018), Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes, IEEE Computer Society, pp. 1091–1100.

Liu, S., Zhang, D. G., Liu, X. H., Zhang, T., Gao, J. X., Gong, C. L. & Cui, Y. Y. (2019), ‘Dynamic analysis for the average shortest path length of mobile ad hoc networks under random failure scenarios’, *IEEE Access* **7**, 21343–21358.

OxfordEconomics (2018), ‘Review of future capacity needs at ireland’s state airports final report for the department of transport, tourism and sport’.
URL: *https://assets.gov.ie/22659/d2cbb36779534741adde4be4f0943a7d.pdf*

Papaioannidis, C., Mademlis, I. & Pitas, I. (2021), Autonomous uav safety by visual human crowd detection using multi-task deep neural networks, Vol. 2021-May, Institute of Electrical and Electronics Engineers Inc., pp. 11074–11080.

Poulos, A., Tocornal, F., de la Llera, J. C. & Mitrani-Reiser, J. (2018), ‘Validation of an agent-based building evacuation model with a school drill’, *Transportation Research Part C: Emerging Technologies* **97**, 82–95.

Prasanth, S., Mathu, T., Santhosh, C. C. & Allwin, F. (2020), ‘Applications of greedy algorithm in human life and the natural world’.

Rozo, K. R., Arellana, J., Santander-Mercado, A. & Jubiz-Diaz, M. (2019), ‘Modelling building emergency evacuation plans considering the dynamic behaviour of pedestrians using agent-based simulation’, *Safety Science* **113**, 276–284.

Song, R., Liu, Y. & Bucknall, R. (2019), ‘Smoothed a* algorithm for practical unmanned surface vehicle path planning’, *Applied Ocean Research* **83**, 9–20.

Still, K., Papalexi, M., Fan, Y. & Bamford, D. (2020), ‘Place crowd safety, crowd science? case studies and application’, *Journal of Place Management and Development* **13**, 385–407.

Stollenwerk, T., Hadfield, S. & Wang, Z. (2021), ‘Toward quantum gate-model heuristics for real-world planning problems’, *IEEE Transactions on Quantum Engineering* **1**, 1–16.

Wan, J. & Chan, A. (2019), Adaptive density map generation for crowd counting, Vol. 2019-October, Institute of Electrical and Electronics Engineers Inc., pp. 1130–1139.

Widera, A. & Hellingrath, B. (2020), 'Trends and future challenges in congestion management prologis-production and logistic intelligent systems view project driver view project', *Conference: 17th International Conference on Information Systems for Crisis Response and Management* pp. 622–636.
URL: *https://www.researchgate.net/publication/342523254*

Zhang, Y., Zhou, D., Chen, S., Gao, S. & Ma, Y. (2016), Single-image crowd counting via multi-column convolutional neural network, Vol. 2016-December, IEEE Computer Society, pp. 589–597.

Zhao, M., Zhang, J., Zhang, C. & Zhang, W. (2019), Leveraging heterogeneous auxiliary tasks to assist crowd counting, Vol. 2019-June, IEEE Computer Society, pp. 12728–12737.

Zhong, X., Tian, J., Hu, H. & Peng, X. (2020), 'Hybrid path planning based on safe a* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment', *Journal of Intelligent and Robotic Systems: Theory and Applications* **99**, 65–77.