Technical University of Berlin
Faculty V of Mechanical Engineering and Transport Systems
Institute of Mechanics
Department of System Dynamics and Friction Physics

Evacuation simulation of the Flying-V using Cellular Automata

Bachelor Thesis

Supervised by: Professor Dr. rer. nat. Valentin Popov
Justus Benad, M.Sc.
Department of System Dynamics and Friction Physics

Submitted by: Raphael Hellmann
Bachelor's degree program: Engineering Science

Place, Date: Berlin, August 14, 2020

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.
This report was not previously presented to another examination board and has not been published.

Berlin, August 14, 2020 Hellmann, Raphael

1. Abstract

Diese Bachelorarbeit behandelt die Entwicklung einer elementaren Evakuierungssimulation des neuartigen Flugzeugkonzepts Flying-V mithilfe von zellulären Automaten. Das Flying-V wurde von Justus Benad, M.Sc., entworfen und wird nun unter der Leitung der Technischen Universität Delft mit Finanzierung von KLM Airlines weiterentwickelt. Zunächst wurden die derzeitigen offiziellen Bedingungen an das Evakuierungskonzept eines Passagierflugzeugs dargelegt und diskutiert. Ein besonderer Schwerpunkt dieser Arbeit lag auf der effizienten und genauen Implementierung der Flugzeuggeometrie. Dies wurde erreicht, indem Bilddateien der Innenraumgeometrie in drei unterschiedlichen Auflösungen erzeugt wurden. Die Bildbearbeitung erfolgte im Freeware-Programm Paint.net, die Implementierung in MATLAB (Version 2019b). Diese Bilddateien wurden in MATLAB eingelesen und ausgewertet. Auf diese Weise konnte die Innenraumgeometrie des Flying-V ausreichend genau dargestellt werden.

Fundamentale Verhaltensregeln von Passagieren in Evakuierungssituationen wurden diskutiert und implementiert. Im Programm kann vorgegeben werden, welche Türen geöffnet sein sollen, wie hoch das Panik-Level der Passagiere ist und mit welcher Auflösung gerechnet werden soll. Alle im Zuge dieser Arbeit geschriebenen MATLAB-Funktionen wurden explizit erläutert. Nach einigen Anpassungen konnten für alle möglichen Türkonstellationen erfolgreiche Evakuierungssimulationen durchgeführt werden. In den Videos zeigten sich natürliche Verhaltensweisen der Passagiere, obwohl nur grundlegende Bewegungsregeln betrachtet wurden. Die Ergebnisse wurden kritisch hinterfragt und alle Möglichkeiten sowie Einschränkungen des Programms diskutiert. Der gesamte Code sowie die der Geometrie zugrundeliegenden Bilddateien und eine Auswahl an Simulationsvideos befinden sich im digitalen Anhang, der dieser Arbeit beigefügt ist. Das Programm soll als Grundlage für eine Weiterentwicklung dienen.

Contents

List of Figures

List of Tables

2. Introduction

Air travel has been a steadily growing market and further substantial growth is expected: The International Air Transport Association (IATA) estimates the total number of annual aircraft passengers will reach 7.3 billion by 2034. This means passenger numbers will more than double in a time-span of 20 years [3]. Due to this, the aircraft manufacturers and airlines are interested in increasing the capabilities and the efficiency of aircraft through ensuring an equal or higher level of passenger safety and comfort. The Flying-V concept addresses both: The innovative outer design in the shape of a V is expected to be highly efficient, while the cabin layout concept is designed to attain a higher level of passenger comfort.

The risk of aircraft accidents will always be present. According to a review of the European Transport Safety Council (ETSC), aircraft accidents can be classified as [7]:

- FATAL or NON-SURVIVABLE (none of the passengers or crew members survive)
- NON-FATAL or SURVIVABLE (all the passengers and crew members survive)
- TECHNICALLY SURVIVABLE (some of the passengers or crew members survive)

It is stated that "Approximately 90 per cent of aircraft accidents are categorised as survivable or technically survivable." [7] One way to avoid fatalities is the prevention of accidents. However, should an accident occur, aircraft occupants should be saved as quickly as possible in order to reduce the number of casualties. According to [7], 40% of fatalities occur in technically survivable accidents. The majority of which occur due to "the effects of smoke, toxic fumes, heat and resulting evacuation problems."

Aside from fire safety and structural improvements, emergency evacuation plays a crucial role in increasing the survival rate of an accident.

2.1. Official requirements for emergency evacuation

In order to receive the operation certification for new aircraft or new seating configurations, a full-scale evacuation trial according to CS 25.803 and Appendix J needs to be performed while being monitored by the appropriate Aviation Authority. This procedure is compulsory for all aeroplanes with a seating capacity of more than 44 passengers, excluding crew members. Concerning EU member states, the responsible Aviation Authority is the European Union Aviation Safety Agency, in the United States, it is the Federal Aviation Administration. The evacuation trial is intended to simulate an aborted take-off. The trial aims to demonstrate "that the maximum seating capacity, including the number of crew members required by the operating rules for which certification is requested, can be evacuated from the aeroplane to the ground under simulated emergency conditions within 90 seconds." [1] Up until 1967, the time limit for evacuation was 120 seconds, however, due to improvements in slide technology, the egress time was adapted to the current value of 90 seconds. Full-scale emergency evacuation trials aim to determine a benchmark "by which the FAA [respectively EASA] can consistently evaluate Evacuation capability using various seating and exit configurations." [10]

In addition to the 90s time limit, the certification process requires numerous other criteria. Regarding the evacuation procedure, these include the following [1]:

- The trial needs to be performed either at night or during simulated darkness.
- Prior to the trial, neither the crew members nor the passengers are informed of particular exits that are to be used.
- 50% of the average amount of carry-on baggage, blankets, pillows etc. needs to be distributed at exit access ways and in the aisles to create obstructions.
- Only half of the emergency exits are allowed to be used.

Furthermore, the participants in the trial need to be a "representative passenger load of persons in normal health" [1]. Some of the specifications concerning the passenger load are the following [1]:

- At least 40% must be females.
- At least 35% must be over 50 years of age.
- At least 15% must be female and over 50 years of age.

2.2. Criticism of full-scale emergency evacuation trials

Full-scale certification demonstrations are costly and expose the participants to significant hazards. The expenses for a trial can easily exceed $1 Million [11]. However, these costs are insignificant compared to the development expenses for a new aircraft. The main criticism is the physical risk for participants. On average, 6% of the participants sustain injuries ranging from cuts and bruises to bone fractures [11]. In October 1991, during the full-scale test of the McDonnell Douglas MD-11, a female volunteer sustained injuries causing permanent paralysis [12].

In addition, the validity of full-scale certification trials is questionable. The trial merely simulates a narrow range of emergency scenarios: "[A]n aborted takeoff at night involving no structural damage, cabin fire, or smoke, for a distinct subset of potential passengers" [11]. Because the participants are aware that they do not face the danger of fire, smoke or inflowing water (during a ditching manoeuvre), this trial fails to consider the impact of panic. Subsequently, the engineers designing the trials are confronted with the dilemma of balancing "realism against the safety of the test participants." [11] This causes the validity of the egress time determined during the trial to be problematic. A tragic example of the possible discrepancy between trial and reality is the Manchester disaster of 1985, which caused 55 people to lose their lives [6]. A Boeing 737's engine caught fire during the take-off process, subsequently, take-off was aborted, and emergency evacuation initiated. The last surviving passenger to escape from the aircraft emerged 5.5 minutes after the aircraft stopped. During the UK certification trial 15 years earlier, the entire passenger load and crew evacuated the aircraft in 75 seconds [12].

The one-time demonstration barely contributes to system optimization, as it provides just a single possibility to detect problems and take corrective options. As stated in the papers [12] and [5], an evacuation simulation produces a "pseudo normal" distribution of egress times when measured repeatedly. According to [12], similar results would be achieved in reality, if the trials were repeated several times. However, due to the certification regulations, only one random data point is assessed that can fall anywhere on this unknown distribution. A more significant value would be a mean egress time calculated throughout many trials, which cannot be gained by following the current demonstration

requirements. Figure 1 graphically demonstrates the arising problem: The assumption that a single trial is capable of indicating the evacuation capability of an aircraft is problematic. The graph illustrates two theoretical sample distributions of egress times for two different airplanes. These distributions could be obtained through numerous full-scale evacuation tests. Based on a single test, it would be possible that an airplane with a "faster" evacuation plan, i.e. a shorter average egress time, fails the test, while the one with the theoretically "slower" evacuation plan passes.

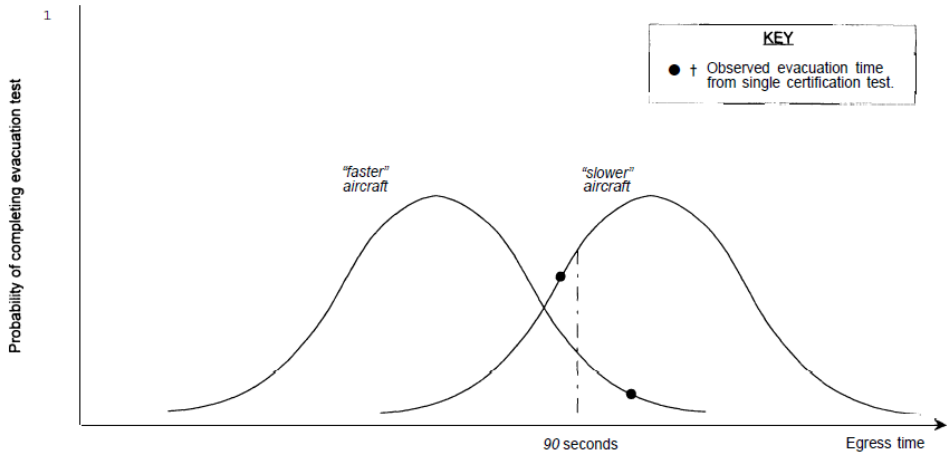


Figure 1: Comparison of Hypothetical Sample Distributions for Aircraft Evacuation Times [11]

2.3. Computational simulation tools

As discussed in the previous section, there are several difficulties in need of addressing to determine the evacuation capability of an aircraft. Computational simulations offer new possibilities and allow developers to approach this issue without exposing anybody to hazards. The generated data can be more reliable than full-scale demonstrations, as the simulation tools are usually run multiple times. Therefore, a distribution of egress times is generated, and an average egress time value can be calculated that is more accurate. In addition, several parameters like operational gates, passenger distribution, behavioural characteristics and others can be altered. This facilitates the optimization process.
It is crucial to note, that appropriate and distinct parameters need to be found in order to simulate the real behaviour of humans in an emergency situation. As stated in [11], "A model is only as good as the parameters which describe the system [...] any evacuation models developed and used will need an extensive program of parameter determination and sensitivity analysis, and an equally extensive validation effort." Due to the preliminary nature of this thesis, only fundamental parameters are going to be considered. This is further examined in Section 4.1.

3. Theoretical Background

3.1. Flying-V concept

The Flying-V is a commercial passenger aircraft concept initially invented and designed by Justus Benad, M.Sc.. The idea is to "efficiently use the volume inside a pure flying wing for commercial passenger transport" [4]. According to [4], a pure flying wing is defined as an object heavier than air with only one lifting surface in the direction of flight and no extra fuselage exposed to the outside. A better understanding of the design can be gained in Figure 2.

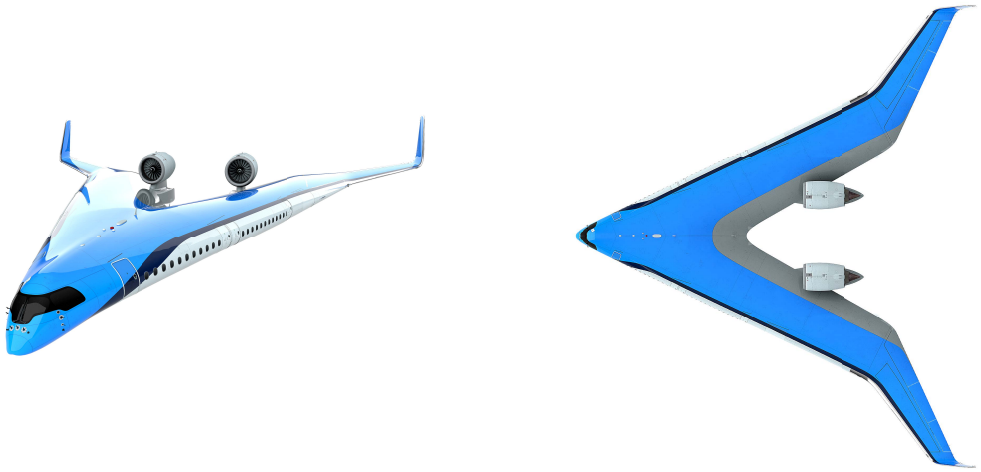


Figure 2: Current design status of the Flying-V concept [8]

Preliminary estimations revealed that this concept might have substantial benefits over state-of-the-art aircraft in terms of efficiency, noise shielding and structural simplicity. Since 2019 the Delft University of Technology is leading the further development process in cooperation with the airline KLM. A 1:20 scale model has already been built, however, there is still a lot of research to be done. An estimation by the project manager at TU Delft, Professor Dr. Roelof Vos, indicates that the Flying-V will not carry passengers before 2040 [14].

One of the areas that need to be investigated is the emergency evacuation process. A preliminary simulation tool using Cellular Automata is presented in the scope of this work.

3.2. Computational model: Cellular Automata

The models used for evacuation simulation are either macroscopic, such as models based on fluid dynamics, or microscopic like Cellular Automata [10]. Cellular Automata (CA) models facilitate efficient analysis of individual passenger behaviour as well as their interaction with other passengers and obstacles.

CA were developed by Stanislaw Ulam and John von Neumann in the 1940s. John H. Conway's "Game of Life", a two-dimensional CA with simple evolution rules, gained great attention in the 1970s due to its stunningly complex behaviour. Another very important contributor to the development process is Stephen Wolfram who published numerous fundamental books and papers of CA theory in the 1980s.

CA are mathematical systems with discrete values in space, time and state. A typical

CA system consists of four components: cells, states, neighbourhood and rules. It can be considered as an artificial universe within which space is partitioned in a uniform grid. The grid cells are the smallest units of the system. Each cell has a finite set of discrete values which evolve synchronously in discrete time steps according to identical rules [15]. Time advances in discrete steps, the evolution rules are applied at each step through which the state of each cell is evolved from the own previous state and those of the neighbouring cells [10]. There are two types of neighborhoods in a two-dimensional square grid: The so-called "von Neumann" neighbourhood (Figure 3a) and the "Moore" neighbourhood (Figure 3b). The number of cells belonging to the neighbourhood varies dependent on the neighbourhood type and the so-called "range" r of the rule [15]: $r = 1$ means that only the nearest neighbors are considered (see Figure 3), for $r = 2$ the adjacent rows and columns are also counted among the neighbourhood, whereas the cell itself is always part of its own neighbourhood. The state of a cell in position i, j is in the following referred to as $a_{i,j}$.

(a) von Neumann neighbourhood (b) Moore neighbourhood

Figure 3: Neighbourhoods in a two-dimensional square grid with $r = 1$

The cells' states evolve according to rules defined as neighbourhood functions [10]. These rules are the driving force behind CA systems and need to be well-defined. As previously mentioned, the state of a cell is updated in every discrete time step according to a rule f that is dependent on the values of cells in the neighbourhood. The value $a_{i,j}^{(t+1)}$ of a cell at time step $t + 1$ in a two-dimensional square grid with Moore neighborhood and $r = 1$ is thus defined as follows:

$$a_{i,j}^{(t+1)} = f\left(a_{i,j}^{(t)}, a_{i-1,j-1}^{(t)}, a_{i-1,j}^{(t)}, a_{i-1,j+1}^{(t)}, a_{i,j-1}^{(t)}, a_{i,j+1}^{(t)}, a_{i+1,j-1}^{(t)}, a_{i+1,j}^{(t)}, a_{i+1,j+1}^{(t)}\right) \quad (1)$$

4. Implementation

4.1. Fundamental rules

This section will discuss the desired overall behaviour of passengers during the evacuation procedure and, deriving from this, textually formulate first rules.

The starting situation in the evacuation process is a fully occupied aircraft with all passengers seated. Firstly, all passengers will attempt to leave the seat aisles and reach their closest main aisle as quickly as possible. This is a fundamental rule of behaviour that needs to be regarded in the simulation. Specific attention must be paid to the first row of seats in Economy and Business Class. If the door in front of them is open, it is not necessary for them to reach the main aisle first, as they can evacuate immediately. This causes a difficulty which will be discussed in Section 6.

Seat jumping, which is a rare phenomenon in aircraft evacuation, will not be considered in the simulation process. Seats, other passengers, and obstacles in the aircraft will be considered insurmountable, similar to walls. This is another fundamental rule that needs to be implemented. Therefore, passengers cannot overtake one another as long as there is not enough space to walk side by side. The space used by each individual passenger, as well as the distance between each of them, will be discussed in further detail in Section 4.3.

After leaving the seat aisles, the passengers will now move along the main aisles in the direction of the nearest open exit. As a result, a distance field must be implemented which respects both the current location of each passenger, as well as the available exits and leads them to the nearest one. As mentioned in Section 2.1, only half of the emergency doors will be opened and prior to the start of the evacuation process, it shall remain unknown which doors have been selected. This causes another requirement for the implementation process: It must be possible to simulate every door combination that fulfills the official requirements. The distance field, therefore, needs to be generated flexibly regarding the input parameters to accurately lead each passenger to their closest door.

On their way to the exits, the passengers do not accidentally enter seat aisles again. This would be possible in the case that grid cells in the seating aisle are closer to the exits than main aisle cells. As a result, it is crucial to prevent passengers from entering into 'dead ends'.

The final aspect of passenger behaviour that needs to be considered during the evacuation process is panic. Up to this point exclusively logical behaviour was assumed for an optimal evacuation. In real evacuation scenarios, panic is always present and it is expected that some of the passengers act rashly and unwisely. It is highly likely that passengers do not pick the optimal direction of movement. An input parameter used to control the probability of acting arbitrarily must, therefore, be introduced.

Under the consideration of these aspects this thesis' preliminary simulation model was created. More detailed explanations will follow in the upcoming sections.

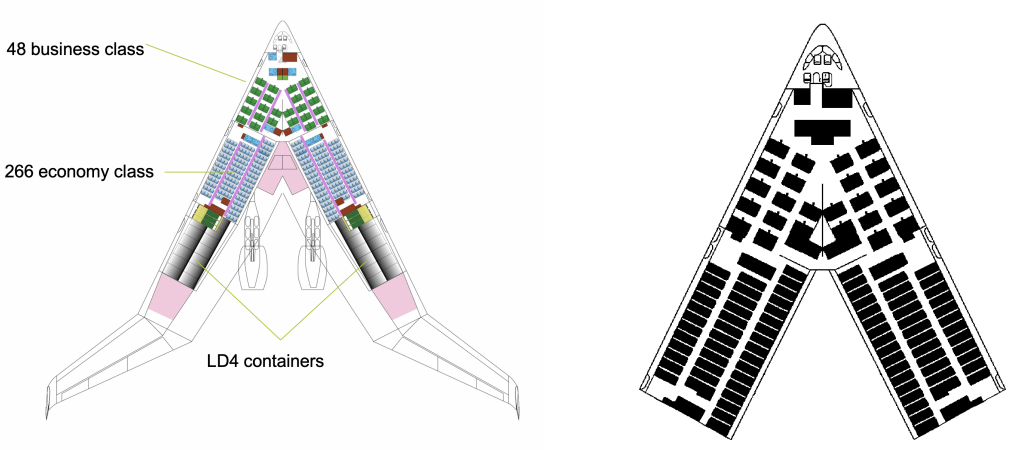
4.2. Implementation of the airplane geometry

The implementation of the simulation tool was done in MATLAB (Version R2019b), a numerical computing environment from the company The MathWorks, Inc..

One major aspect of this thesis was the efficient and accurate implementation of the airplane geometry. The V-shape of the Flying-V aircraft contains challenges: The diagonal walls and seat rows are not easily projected on orthogonal matrices. Especially because

the wings' sweep angle of 63° [9] differs from the 45° diagonal. The aim of the proposed idea of geometry implementation is to be as generally valid as possible. It should be easily transferable to other airplane geometries.

The idea is to import a detailed image file of the airplane geometry in the program and to automatically extract the required information. For generating and adapting image files the freeware raster graphics editor Paint.net was used. With this program, a 'pencil sketch' of the existing interior concept was automatically created and then manually adapted. Only the passenger cabin was extracted since the evacuation process only occurs in this area. Figure 4 gives a better understanding of the created 'pencil sketch':



(a) Interior concept of the Flying-V [9] (b) Manually adapted 'pencil sketch' of the passenger cabin

Figure 4: Extracting the passenger cabin geometry

In the process of automatically generating a 'pencil sketch', the program colours each pixel black if its initial colour is not white (with a specific tolerance). A black and white picture is created. Due to the limited resolution of the initial picture, the generated picture is not sufficiently accurate. Thus, it must be manually adapted to ensure that there is no gap in the walls or seats and that the outlines match the real shape as precisely as possible. The number of pixels in the 'pencil sketch' determines the size of the matrices in MATLAB: When importing the image file in MATLAB, each pixel is one entry in the matrix. The colour of the pixel determines the value of the entry. This means that an image file with more pixels leads to a greater matrix, which, in turn, leads to higher accuracy in the simulation. An input parameter which exactly defines the size of the field is not possible due to the manual adaptation. The preliminary solution is to generate and adapt three 'pencil sketches' with various pixel densities to compare the accuracy of the simulation afterwards. An input parameter k is introduced to choose between the three available image files for the simulation.

As mentioned in Section 4.1, attention must be paid to the seat aisles. To handle the cells in the seat aisles separately, these areas must be easily identifiable. The seat aisle areas in the sketch were therefore manually coloured in red. In addition to the seat aisles, the door cells also need to be identified. The Flying-V has four type A doors per side, one at the front end of each wing, two in the middle and one at the rear end. According to [1], each type A door must have a double slide, which means that two passengers can exit simultaneously. Two cells per door are therefore marked as 'exit cells' by colouring

them yellow.

Finally, passengers can be placed. Due to a relatively high pixel density in all of the three generated image files, one cell in the grid does not correspond with the dimensions of one passenger. This problem is solved by introducing 'passenger centres': For each passenger, one cell is coloured. This cell represents the centre of the passenger. The calculation process of the actual dimensions of the passengers is explained in Section 4.3. As there are 48 seats in the Business Class and 266 seats in the Economy Class, a total of 314 passenger centres are placed for a fully occupied Flying-V. In front of each passenger seat, one cell of the seating aisle is thus manually coloured in blue. Special attention is paid to the passenger seats in the first rows of the Business and Economy Class: As explained in Section 4.1, the passengers in these rows do not have to reach the main aisle first, should an emergency scenario occur within which the door directly in front of them is used to evacuate. The passenger centres belonging to these positions are coloured in green for these to be considered separately. A more detailed instruction of the entire adaptation process can be found in the appendix.

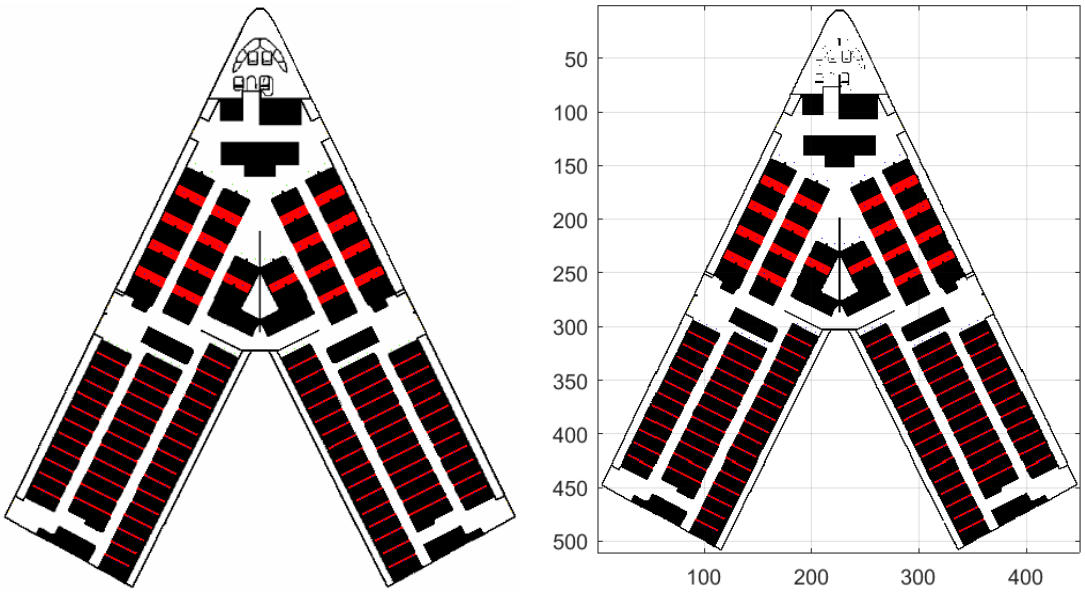
After manually adapting the image file to this state, a PNG file is generated. The following procedure is applicable for all three pixel densities. To store the pixel information in the PNG format, a bit depth of 8 bit is used. This format is limited to 256 different colours, however, this is sufficient for this application. As a result, the colour values are stored as a number between 0 and 255 defining the position on the colour palette. Now the PNG file can be transferred to MATLAB and is automatically imported. The depth of 8 bits is beneficial due to the fact that after importing the PNG file in MATLAB the generated array only has one layer. When importing the PNG file MATLAB generates a matrix (from now on referred to as 'initial matrix') the same size as the number of pixels in each direction. The pixel colour defines the value of the corresponding matrix entry. Now the desired information can be extracted from this matrix. Different geometry matrices are preassigned and all match the size of the initial matrix. The *WALL* matrix serves as an example: This matrix shall have the value 1 in each cell where the initial matrix has the value for black. Everywhere else it shall have the value 0. The generated matrix contains the locations of each cell with an obstacle: This means walls, seats and every other kind of obstacle that cannot be crossed. For the *DOOR* matrix, the initial matrix is analyzed for each entry that matches the value of the used shade of yellow. According to the input vector *open_doors* (given by the user), only the desired exit cells are stored in *DOOR*. The same procedure of reading out the values is conducted for the rest of the geometry matrices: *PASSENGER*, *PASSENGER_AISLE*, *AISLE*, *CLOSED_DOOR*. Table 1 gives an overview of the information saved in these matrices.

Matrix name	Stored information
WALL	Location of obstacles
DOOR	Location of open doors
CLOSED_DOOR	Location of closed doors
AISLE	Seat aisle area
PASSENGER	Current location of passenger centres
PASSENGER_AISLE	Location of passenger centres in seat aisle area

Table 1: Geometry matrices

The matrix *PASSENGER_AISLE* is only used for an accurate start configuration plot. The reason for colouring the passenger centres in the first rows green instead of blue lies in the double information of passenger centres in the seat aisle area: The cells in the seat aisles occupied by passenger centres must be counted both to *PASSENGER* and to *AISLE*. The cells in the first rows occupied by passenger centres shall not be counted to *AISLE*, as these areas are not defined as seat aisles. Therefore, they needed to be clearly identifiable.

After extracting and storing this information, the airplane geometry can be precisely displayed in a plot. Following this, a colourmap is defined and the specific matrices are multiplied by scalar factors to match the same colours as in the PNG file (for aesthetic reasons). Figure 5 shows the comparison of the PNG file and the MATLAB plot.



(a) PNG file (511x450 pixels) (b) MATLAB plot (511x450 cells)

Figure 5: Rendering of airplane geometry in MATLAB

Due to the high pixel density and the colouring of single cells, the door cells and the passenger centre cells are hardly visible. The location of passengers will be visually clearer in Section 4.3. As previously discussed, *PASSENGER* stores the location of the entire number of passenger centres. Hence, there is no longer a difference between the passenger centres in the front rows and the others. As a result, all passenger centres are blue in the MATLAB plot.

The geometry matrices are stored in the 'sparse' format in MATLAB due to the high number of zero values. As long as the same colours are used for adapting the image files, this procedure of implementing the airplane geometry is easily transferable to any other geometry.

4.3. Passenger dimensions

To address the fact that one cell in the grid does not match the size of one passenger, a passenger radius is introduced. This radius defines the area demanded by one passenger and shall surround each passenger centre. According to [2], the area taken by one person in a dense situation corresponds to 0.4m by 0.4m. It was assumed that a circular area

would be more advantageous to avoid passengers getting caught on stepped and cornered wall sections. Therefore, a radius of $r = 0.22$m was chosen to approximately create the same surface area. The cells around each passenger centre that are within each passenger's radius thus also get the value 1 in *PASSENGER*. The exact implementation of finding the correct cells around each passenger centre is explained in Section 4.4.2. Figure 6 shows the final geometry plot for the start configuration obtained with the adapted *PASSENGER* matrix.

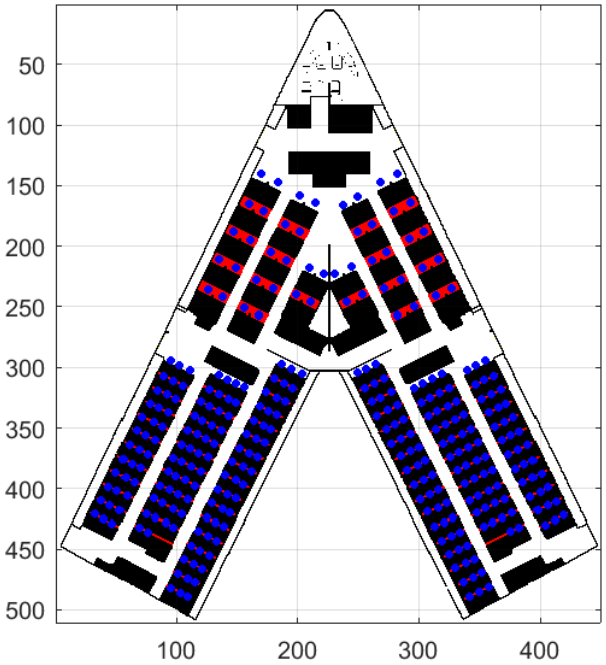
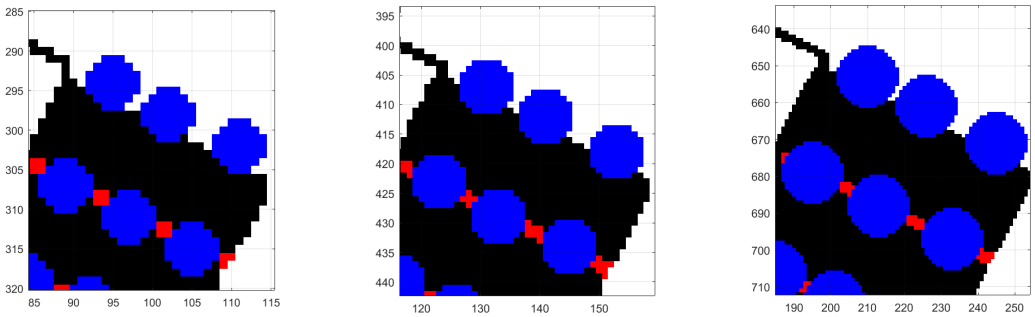


Figure 6: MATLAB plot of airplane geometry with passenger radii (511x450 cells)

It is crucial to note, that in this tool the passenger radius is not considered a solid wall. It is possible to enter the passenger radius with consequences for the movement speed (explained in Section 4.4). Therefore, a passenger can approach an obstacle or enter another passenger's radius. It is, however, impossible to step on an obstacle cell with a passenger centre. Collisions of passengers, which means the collision of two passenger centre cells, will be dealt with in Section 4.4. The centre cells can thus be seen as solid kernels. Inevitably, the passenger radii become more circular with increased accuracy. Figure 7 demonstrates this with the three available resolutions.



(a) 511x450 cells (b) 707x623 cells (c) 1135x1000 cells

Figure 7: Passenger radii in different accuracies

4.4. Simulation Process

In this section, the simulation tool is explained in detail. A flowchart will show the general succession of steps and will give an overview over the created MATLAB functions. Afterwards, each function will be explicitly examined.

Firstly, the fundamental properties of a CA tool need to be defined. The simulation is run on a square grid and the chosen neighborhood is the Moore neighborhood (see Figure 3b). This enables diagonal movements which should be possible for a natural walking behaviour of the passengers. There are two possible ranges: $r = 1$ and $r = 2$. The idea is to distinguish between 'fast' and 'slow' passengers. Each passenger whose radius overlaps with another passenger's radius or obstacle cells is a slow passenger. Every other passenger stays a fast passenger. This represents the fact of limited freedom of movement and is explained in detail when the specific MATLAB function is presented. A fast passenger - more precisely: a fast passenger centre cell - thus has a Moore neighbourhood with $r = 2$ and a slow passenger has a Moore neighbourhood with $r = 1$. The neighbourhood will in the following be called 'radius of movement'. The fact that the distance between centre cells and diagonal cells is greater than between centre cells and horizontal or vertical cells is neglected due to the preliminary character of this tool. The movement of passengers is realized by relocating the passenger centre cells. For the plot and for the distinction between 'slow' and 'fast' passengers, the radii are taken into account. However, the main operations are implemented by working with the passenger centre cells. As mentioned in Section 4.2, the current locations of these cells are read out of the *PASSENGER* matrix. Figure 8 shows the general steps in the simulation process. The boxes with dotted framing contain information about the stage of the simulation, whereas the boxes with solid framing name the called MATLAB functions with their input and output parameters. To create a clear structure in the program code, there is a main function which proceeds similarly to the steps in the flowchart. Each other function is called by the main function. The naming of variables in the entire code follows the structure shown in Table 2.

Name	Meaning
TEST	Matrix with size of the entire grid
TEST_ind or *Test*	Matrix or Array with different size
test or *test_ind*	Vector or Scalar

Table 2: Naming structure in the code

The process starts with entering the input parameters in the main function. Firstly, the vector *open_doors* needs to be defined. This 8x1 vector determines the open doors. The eight doors of the aircraft are numbered clockwise, beginning in the lower left corner. 1 means open, 0 closed. According to [1], only 50% of the exits are allowed to be open. Hence, *open_doors* can contain any combination of four ones and four zeros. k determines the accuracy in the simulation process. The available resolutions are listed in Table 3.

k	Resolution
1	511x450 cells
2	707x623 cells
3	1135x1000 cells

Table 3: Resolution of the simulation according to k

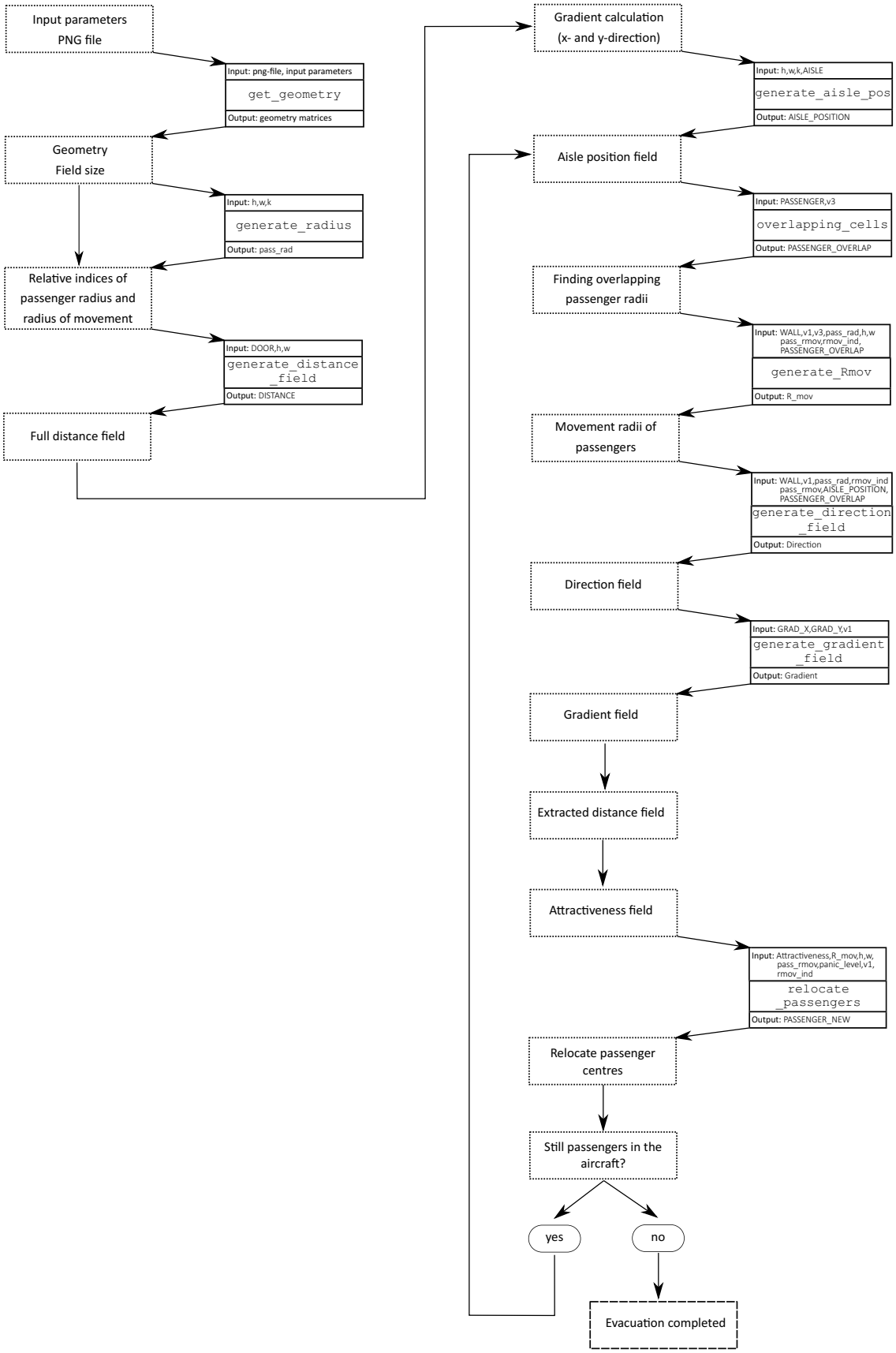


Figure 8: Flowchart of the simulation process

The higher the accuracy, the greater is the calculation time. k should therefor always be chosen with a 'cost-benefit' thought in mind. The last input parameter is *panic_level*. This parameter is the probability for each passenger to behave arbitrarily. For $panic\_level = 0$, all passengers behave exclusively rational and always decide for the best direction of movement. For $panic\_level = 1$, all passengers behave exclusively arbitrarily and choose their direction of movement randomly. This parameter represents the influence of panic in the evacuation process.

Now the simulation process begins. Before the MATLAB functions are explained specifically, the general steps need to be outlined. k determines which of the available PNG files is chosen. According to this PNG file, the geometry matrices are generated by letting MATLAB read out the required information. Now the entire airplane geometry and the initial locations of the passenger centre cells are defined. By analyzing the size of any of the geometry matrices, the size of the field is determined: The number of rows defines the height h of the field, the number of columns the width w. Subsequently, the passenger radius needs to be clarified. As mentioned in Section 4.3, the passenger radius is $r = 0.22$ m. This must be ensured for each accuracy. The real dimension of the airplane is always the same, the accuracy only determines the number of cells in the field. The higher the accuracy, the greater is the number of cells which are allocated to each passenger radius. Due to the fact that, in this tool, each passenger has the same passenger radius, relative indices can be used and are stored in the vector *pass_rad*. This facilitates applying the passenger radius to each passenger centre cell and will be explained in further detail in Section 4.4.2. Now the radius of movement is defined: As previously mentioned, the maximum radius of movement shall be a Moore neighbourhood with range $r = 2$ around the centre cell. Therefore, the radius of movement is generally (in all three accuracies) defined as a 5x5 block with the centre cell in the middle. 'Fast' passengers can reach the outer cells of this block in one time step, 'slow' passengers only the inner cells. Depending on the velocity of the passengers, the cells in the 5x5 block are adapted (explained in Section 4.4.6). To facilitate the application of this radius to each passenger centre cell, relative indices are used again. To guarantee the same maximum velocity of the passengers in all three accuracies, the time step is adapted according to the cell size.

The first field, which influences the passenger movement, is the distance field *DISTANCE*. According to the naming structure, this matrix has the size of the entire field. It contains the distance of each cell to the respectively closest open door. In the case that passenger centre cells are very far away from the next open door, the distance values in the adjacent cells do not differ significantly. A clear decision for the movement could therefore be problematic. For this reason, additional fields are generated: the distance gradient fields *GRAD_X* and *GRAD_Y*. *GRAD_X* contains the numeric partial derivative of $(-DISTANCE)$ in x direction. The reason for calculating the gradient values of $(-DISTANCE)$ is to let the gradient vector point in the direction of decreasing distance values. For *GRAD_Y*, the numeric partial derivative in y direction is calculated. With both fields combined, the resulting gradient vector can be calculated for each cell. The directions of these gradient vectors influence the passengers' movement choice and shall ensure a clear decision in the cases mentioned above.

As discussed in Section 4.1, all passengers (except in some cases the ones in the first rows) must leave the seat aisles first before they can approach the exits. Therefore, the field *AISLE_POSITION* is created. It represents the position in the seat aisles: the greater the distance between cell and main aisle, the higher is the value. The cells' values thus decrease the closer they are to the main aisles. This forces the passengers to leave the

seat aisles first. For seat aisles with main aisles to both sides, the values are the highest in the middle and decrease to both sides. *AISLE_POSITION* is thus another field which influences the passengers' movement choice.

After generating these fields, the time loop begins. The current location of passenger centre cells is stored in the vector *v1*. It saves the indices of cells in *PASSENGER* with value 1. Now *PASSENGER* is adapted by applying the passenger radius around each passenger centre cell with the relative indices stored in *pass_rad*. In the case that passengers stand close to each other, their passenger radii overlap. It is important to find out whose passenger radii overlap for dividing the passengers in 'slow' and 'fast'. For this reason, the matrix *PASSENGER_OVERLAP* is created: It allocates each cell with the number of passenger radii, in which this cell occurs. Figure 9 gives a better understanding of the difference between *PASSENGER* and *PASSENGER_OVERLAP*. It shows the overlapping passenger radii of the same two passengers in both matrices. The passenger centre cells are marked with a red frame.

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

(a) Overlapping passenger radii in *PASSENGER* ($k = 1$)

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	2	1	1	0	0	0	0
1	1	1	1	1	2	2	1	1	1	0	0	0
1	1	1	1	2	2	2	1	1	1	1	0	0
0	1	1	1	2	2	1	1	1	1	1	0	0
0	0	1	1	2	1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Overlapping passenger radii in *PASSENGER_OVERLAP* ($k = 1$)

Figure 9: Difference between *PASSENGER* and *PASSENGER_OVERLAP*

The next step is to generate the radius of movement for each passenger. As previously mentioned, the movement radius is initially a 5x5 block around each passenger centre cell. After dividing the passengers into 'slow' and 'fast', the movement radii are adapted. Hence, it needs to be clarified which passenger's radius overlaps with another passenger radius or with an obstacle. If one or both of the cases are true, this passenger is 'slow'. If not, this passenger stays 'fast'. This is simultaneously checked for all passengers. For the 'fast' passengers, the inner cells (except the middle cell) are allocated with NaN. This means 'Not a Number' and facilitates the movement decision later in the process as these cells are considered unreachable. For the 'slow' passengers, the outer cells are allocated with NaN. Figure 10 shows the resulting movement radii.

1	1	1	1	1
1	NaN	NaN	NaN	1
1	NaN	1	NaN	1
1	NaN	NaN	NaN	1
1	1	1	1	1

(a) 'Fast' passenger

NaN	NaN	NaN	NaN	NaN
NaN	1	1	1	NaN
NaN	1	1	1	NaN
NaN	1	1	1	NaN
NaN	NaN	NaN	NaN	NaN

(b) 'Slow' passenger

Figure 10: Adapted movement radii

The principal structure is now set: The cells in the 5x5 block with the passenger centre cell in the middle are considered as possible candidates for the movement decision. The NaN cells in the movement radius of each passenger are unreachable. The movement radius is additionally adapted by analyzing which of the cells, that are as yet rated as reachable, are obstacle cells. This is done by allocating each cell with NaN, which occurs in the movement radius of any of the passengers and which has the value 1 in the *WALL* matrix. This represents the fact that passenger centre cells cannot occupy obstacle cells (passengers cannot enter walls or other obstacles). There is one final adaptation to be done: In the case that two passengers decide for the same cell (collision), it needs to be ensured that one of them can return to his/her initial centre cell. Therefore, all cells in the movement radii of the passengers, which are currently the centre cell of another passenger, also need to be allocated with NaN. After these adaptations, only the reachable cells remain in the movement radii. Now it needs to be found out which of the remaining cells is the best to come closer to the exits.

The cells in the movement radius are rated with attractiveness values. In this tool, the smallest value is the most attractive. These attractiveness values are generated by linear combination of three different influence fields: *Distance*, *Gradient* and *Direction*. As previously mentioned, the current position in the seat aisle must have a distinct influence on the movement decision. Therefore, the field *Direction* is generated: In addition to the position in the seat aisle (read out of *AISLE_POSITION*), *Direction* considers the proximity to obstacles and other passengers. Cells which lead closer to obstacles or other passengers, and thus would make the passenger 'slow', are allocated with less attractive values. The exact values depend on the number of cells which would overlap with obstacle cells or other passengers' radii if the passenger decided for this specific cell. The field *Gradient* is generated by analyzing the direction of the distance gradient vector of each passenger centre cell. The x and y components of these vectors are read out of *GRAD_X* and *GRAD_Y*. Now the vector's direction is compared with the direction of the cells in the movement radius. The better the direction of the cell (seen from the centre cell) matches with the direction of the gradient vector, the more attractive is the value. If the directions of vector and cell differ more than 22.5°, the cell gets a "mismatch" value. As a result, only the cells pointing in approximately the same direction as the vector get reasonable values. This shall ensure a clear decision in the case that the cell's distance values are not sufficiently different. *Distance* is generated by simply extracting the values of the cells in the movement radius of each passenger out of *DISTANCE*.

By linear combining these fields according to Equation (2), the final field *Attractiveness* is generated.

$$Attractiveness = t \cdot (s \cdot Distance + (1 - s) \cdot Gradient) + (1 - t) \cdot Direction \quad (2)$$

t and s are adjustment parameters. Their values are heuristically found and discussed in Section 5.4. This field contains the resulting attractiveness values for all cells in the movement radii. Each cell in *Attractiveness* that is considered unreachable is now overwritten with NaN. As mentioned earlier in this section, *panic_level* determines the arbitrary behaviour of the passengers. With the probability $(1 - panic\_level)$ the passengers decide for the most attractive of the remaining cells. With the probability *panic_level* they decide randomly for one of the remaining cells. The movement of passengers is done by setting the value of the chosen cell to 1 and all other cells in the movement radius of this specific passenger to 0. In this way the passenger centre cell has been moved. There

are two scenarios that need to be regarded: In the case that two passengers decide for the same target cell, which would cause a collision, one of them is randomly chosen and relocated to his/her initial centre cell. The other scenario happens when two or more cells in the movement radius of one passenger have the exact same attractiveness value. If this value is the most attractive in the movement radius, all of these cells are chosen to be new passenger centre cells. This means that passengers would be created out of nowhere. This problem is solved by randomly choosing one of the new passenger centre cells in the movement radius and deleting the other ones, so that there is always only one passenger centre cell per movement radius.

With the relocated passenger centres a new *PASSENGER* matrix is created. The last step of every time loop iteration is to clear the exit cells: If passenger centre cells are standing on exit cells, they have reached the door and can leave the aircraft. These cells in the new *PASSENGER* matrix are therefore set to 0. After this adaptation, the next time step begins with finding the new locations of passenger centre cells. The time loop is iterated until all passengers have left the aircraft.

4.4.1. get_geometry

Input: *open_doors*, k
Output: Geometry matrices

This function generates the geometry matrices *WALL*, *AISLE*, *DOOR*, *CLOSED_DOOR*, *PASSENGER* and *PASSENGER_AISLE*. Depending on the parameter k, it imports one of the available PNG files. According to the colour values of the pixels, it extracts the information on the aircraft's geometry. All of the matrices contain only 1 and 0 values. 1 means that this cell is occupied in the specific matrix, 0 that it is empty. 1 in a *WALL* matrix cell means, for example, that there is some kind of obstacle. The geometry matrices are stored in 'sparse' format to reduce the calculation effort in the following steps. The only matrices that need to be adapted are *DOOR* and *CLOSED_DOOR*: In the PNG file, all possible exit cells are marked with a specific shade of yellow. Depending on *open_doors*, only the chosen doors shall be open. For this reason, *open_doors* is analyzed in this function and the exit cells of doors, that are determined to be closed, are overwritten with 0 in *DOOR*. The exact opposite happens with *CLOSED_DOOR*: the cells of doors, that are determined to be open, are overwritten with 0. *CLOSED_DOOR* is only used to highlight the closed doors in the plot.

4.4.2. generate_radius

Input: h, w, k
Output: *pass_rad*

This function generates the relative indices around each passenger centre cell which are within the defined passenger radius of $r = 0.22$m. Since every passenger has the same radius, the relative indices only need to be calculated once. k determines the cell size, whereas height and width are identical (square cells). This length, in the following called l, has been approximately read out of the geometry plot for each available accuracy. According to k, the respective value for l is chosen. Now a two-dimensional mesh with step size l in both directions is generated with the centre point defined as (0|0). With the x and y values, the Euclidean distance of each grid point to the centre cell is calculated.

The grid points within $r = 0.22$m define the passenger radius. Now the relative indices of these cells referring to the centre cell are determined and stored in the vector *pass_rad*. Hereby linear indexing is used: This means that the position of the cell is defined by a single index and not by row and column. With this vector it is possible to access the passenger radius cells of any passenger just by setting the index of the specific passenger centre cell as the new reference point.

4.4.3. generate_distance_field

Input: *DOOR*, h, w
Output: *DISTANCE*

This function generates a static field which contains the distance values between each grid cell and the respectively closest door. Due to the fact that the position of the open doors and the grid cells in the field do not move during the simulation process, it only needs to be generated once. As mentioned in 4.2, there are two exit cells per open door. Four of the eight doors are open during the simulation process, and since the distances between each field cell and each exit cell must be calculated, there are eight different distance values per field cell. The distance calculation is again done by generating two-dimensional grids with the specific exit cell defined as the point $(0|0)$. The Euclidean distance of each field cell to the specific exit cell is calculated with the x and y coordinates. Hereby, the step size of the grid does not have to be l since the distance values do not have to be conform to the actual airplane dimensions. The difference in distance values of cells in the movement radius is important for the passenger's decision, not the actual distance. The grid's step size is, for simplicity reasons, thus set to 1. From the eight possible distance values for each field cell, always the smallest value is chosen and stored in *DISTANCE*.

4.4.4. generate_aisle_pos

Input: *AISLE*, h, w, k
Output: *AISLE_POSITION*

This function generates a field which rates the seat aisle cells according to their distance to the closest main aisle. This ensures that the passengers leave the seat aisles first before they start to approach the doors. The aim is to allocate seat aisle cells which are far away from the main aisle with high values and letting the values in each seat aisle decrease in the direction of the main aisle. This reveals a difficulty: In the case that the main aisle is on the right side, the values must decrease to this side. In the case that the main aisle is on the left side, the values must decrease to the left side. And in the case that there are main aisles to both sides, the values must decrease to both sides. This needs to be ensured for both cabin sections. The solution in this tool is to define fourth-degree polynomials for each cabin section which have their double zeros in the main aisles. Going from left to right in each cabin section, this ensures that the values primarily decrease until the zero is reached in the left main aisle, then increase until they reach the maximum point in the middle of the middle seat row, and then decrease until they reach the second zero in the right main aisle. From there on, they increase again along the right seat row. As mentioned, this has to be done separately for each airplane half to achieve the desired behaviour of values. Another challenge is that the main aisles run

diagonally, therefore the zeros of the polynomials also have to be shifted. This is realized by generating a matrix of values - called *MESH* - on which the fourth-degree polynomials are evaluated. The values in *MESH* start with 0 in the middle of the first matrix row and increase to left and right with step size 1. *MESH* has the same size as the geometry matrices, so each field cell is represented by one value. Every second row the values are shifted by one position to the left in the left half and by one position to the right in the right half. This is done in order to let the values approximately follow the diagonal of the main aisles. It would also be possible to define a fourth-degree polynomial for each row to address the fact that the zeros must shift according to the main aisle diagonals. Then the *MESH* field's values would not have to be shifted to left and right. But this would be unnecessarily complicated to implement compared to the proposed procedure.

The fourth-degree polynomials are created with the factored form by reading out the *MESH* values of the cells where the double zeros shall be located. Since the main aisles in the Business and the Economy Class do not lie on the same diagonals, a polynomial must be created for each class. The left and the right aircraft half, however, are constructed identically, so the same polynomials can be used for both cabin sections since *MESH* is symmetrical to the centre longitudinal line. Figure 11 gives a better understanding of the idea: It schematically shows the course of one of the polynomials in one row in one aircraft half. The fact that the values at the outer positions are higher than in the middle is not a problem since only the relative values are important for the passenger's choice.

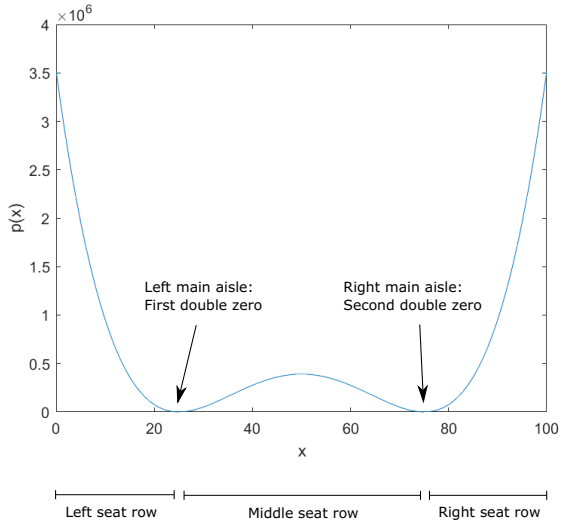


Figure 11: Schematic desired course of the fourth-degree polynomial

The fully defined fourth-degree polynomials are evaluated on the entire *MESH* field. Since these values shall only be existing in the seat aisles, the generated field is pointwisely multiplied with the matrix *AISLE*. *AISLE* only has ones in the seat aisle cells and zeros everywhere else, so only the values in the seat aisles are extracted and the other cells' values are set to 0. This ensures that *AISLE_POSITION* only influences the passengers' behaviour in the seat aisle cells.

4.4.5. overlapping_cells

Input: *PASSENGER*, *v3*
Output: *PASSENGER_OVERLAP*

This function generates a field which locates overlapping passenger radii. It also contains information on how many passenger radii overlap in the specific cells. The cells within the passenger radius around each passenger centre cell are already set to 1 in *PASSENGER* when it is handed over to this function. The vector *v3* contains the indices of the passenger radius cells of all passengers. These are sorted by passengers, so if there are 42 cells in each passenger radius, then the first 42 values in *v3* belong to the first passenger, the next 42 values to the second passenger, and so on. To find out where passenger radii overlap, it simply needs to be clarified which field cells are represented in more than one passenger radius. Therefore, *v3* is checked for values that occur more than once. If this is the case, then the function counts how often this specific value is represented in *v3*. The number of appearances defines the number of overlapping passenger radii on this specific cell. So *PASSENGER_OVERLAP* is generated by overwriting the values of cells in *PASSENGER*, that are represented in more than one passenger radius with the number of appearances in *v3*.

4.4.6. generate_Rmov

Input: *WALL*, *v1*, *v3*, *pass_rad*, *h*, *w*, *pass_rmov*, *rmov_ind*, *PASSENGER_OVERLAP*
Output: *R_mov*

This function generates an array with n layers, where n represents the number of passengers in the aircraft. Each layer contains the Moore neighbourhood with range $r = 2$ (5x5 block) of one specific passenger centre cell. As mentioned in section 4.4, each layer has to be adapted according to the velocity of the passenger, surrounding obstacles, and other passengers. Firstly, the passengers need to be divided into 'slow' and 'fast'. *v3* is thus reshaped into a matrix A, in which each column contains the indices of passenger radius cells of one passenger. Now a summation matrix of *WALL* and *PASSENGER_OVERLAP* is generated. This summation matrix is evaluated on the indices saved in A. The resulting matrix is added up column-wise and the sum values are saved in a row vector. This row vector contains n entries, in which each entry represents the sum of all *WALL* and *PASSENGER_OVERLAP* values in the passenger radius cells of one specific passenger. If this value differs from the number of cells in the passenger radius, it means that either there are overlapping cells or obstacle cells (or both) in the passenger radius of this passenger. This means that this passenger is close to a wall or another passenger, and thus needs to be moving slowly. If the value is identical to the number of passenger radius cells, it means that there are no overlapping cells or obstacles within the radius: This passenger thus can stay fast. According to this classification, each layer in *R_mov* is allocated with the correct movement radius (see Figure 10) for this passenger.

It now needs to be ensured that obstacle cells in the movement radii are unreachable. The vector *rmov_ind* contains the indices of all cells in the movement radii, again sorted by passengers. *WALL* is now evaluated on the indices stored in *rmov_ind* and the resulting vector is reshaped to match the 5x5xn dimension of *R_mov*. Each layer of this resulting array, called *Wall_val*, now contains the *WALL* values within the radius of movement of the passenger. The order of passengers is identical to the one in *R_mov*. So *R_mov* can

simply be compared to *Wall_val*: The cells in *R_mov* which have a 1 in *Wall_val* are overwritten with NaN.

The final adaptation in *R_mov* is to ensure that a passenger's centre cell is unreachable for all other passengers. A temporary passenger matrix with size hxw is generated, where only the passenger centre cells are allocated with 1, the other cells stay 0. This matrix is evaluated on the indices stored in *rmov_ind*. The resulting vector is again reshaped into a 5x5xn array. Each layer in this array now contains the 5x5 block around the centre cell of this passenger.The cell in the centre of the block is set to 0 in each layer. The remaining cells with value 1 in the array are passenger centre cells which are in the movement radius of other passengers. These cells are overwritten with NaN in *R_mov*.

4.4.7. generate_direction_field

Input: *WALL*, *v1*, *pass_rad*, *pass_rmov*, *rmov_ind*, *PASSENGER_OVERLAP*, *AISLE_POSITION*
Output: *Direction*

This function generates one of the influencing arrays for passenger decision. *Direction* has the same dimension as *R_mov*, i.e. 5x5xn (n is again the number of passengers inside the aircraft). Firstly, the vector *rmov_ind* is reshaped into a matrix, so that each column contains the indices of cells in the movement radius of one specific passenger. This matrix is transposed, so that now each row contains the indices of movement radius cells of one passenger. This matrix is called *R*. The order of indexing in the movement radius begins in the upper left corner of the 5x5 block and follows the linear index up to the lower right corner. The cells in the 5x5 block are now seen as movement directions and will in the following be called according to Figure 12. Direction $R13$ is equivalent to not moving.

R1	R6	R11	R16	R21
R2	R7	R12	R17	R22
R3	R8	R13	R18	R23
R4	R9	R14	R19	R24
R5	R10	R15	R20	R25

Figure 12: Naming of the movement directions

The first column of *R* thus contains the indices of all R1 cells, sorted by passenger. To find out how the proximity to walls and other passengers, as well as the position in the seat aisle, changes for every possible direction, the values within the shifted passenger radius need to be analyzed. For this reason, a matrix is generated which contains the indices of passenger radius cells around each of the entries in *R*. This matrix is called *ALL_ind*. A summation matrix is generated by adding up *WALL*, *AISLE_POSITION* and *PASSENGER_OVERLAP*. Now this summation matrix is evaluated on each entry of *ALL_ind*. The result is a matrix the same size as *ALL_ind*, but instead of the indices it now contains the values of the summation matrix in each cell represented in *ALL_ind*. Moreover, all values which belong to the same passenger radius are added up. The resulting matrix, called *SUM_values*, has the same size as *R* again, but each entry is the sum of values in the passenger radius of this movement direction. The first entry in the first row of *SUM_values*, for example, expresses the sum of values in *WALL*, *AISLE_POSITION* and *PASSENGER_OVERLAP* in the passenger radius, if passenger number 1 decides to

move to R1 (in his/her movement radius). The lower the sum, the more attractive is the direction for the passenger. *SUM_values* is then reshaped into a 5x5xn array to be consistent with *R_mov*. Each layer now contains the sum values for each possible movement direction of the passenger.

4.4.8. generate_gradient_field

Input: *GRAD_X*, *GRAD_Y*, *v1*
Output: *Gradient*

This function generates the second influencing array for the movement decision. For each passenger, it needs to be clarified which of the movement directions matches the gradient vector best. Therefore, the position of the movement directions, viewed from the respective centre cell, is defined by the angle between the connecting line and the horizontal axis. Since the movement radius size is identical for each passenger, these angles only need to be calculated for one standard 5x5 block. The movement directions are characterized by their coordinates displayed in Figure 13a. With these coordinates, the cell angle d of each movement direction is calculated according to

$$d = \left| tan^{-1}\left(\frac{y}{x}\right) \right|. \tag{3}$$

These values are rounded to zero decimal places and saved in a 5x5 matrix called *Degree*, seen in Figure 13b. The algebraic signs of the coordinates in Figure 13a are stored in matrices for the comparison procedure.

x

(-2|-2)	(-1|-2)	(0|-2)	(1|-2)	(2|-2)
(-2|-1)	(-1|-1)	(0|-1)	(1|-1)	(2|-1)
(-2|0)	(-1|0)	(0|0)	(1|0)	(2|0)
(-2|1)	(-1|1)	(0|1)	(1|1)	(2|1)
(-2|2)	(-1|2)	(0|2)	(1|2)	(2|2)

y

(a) Cell coordinates of a standard 5x5 block

45°	63°	90°	63°	45°
27°	45°	90°	45°	27°
0°	0°	NaN	0°	0°
27°	45°	90°	45°	27°
45°	63°	90°	63°	45°

(b) *Degree*

Figure 13: Cell angle calculation

Now the actual gradient vectors are analyzed. The x and y components of these vectors are found by extracting the values of the current passenger centre cells out of *GRAD_X* and *GRAD_Y*. The angle between gradient vector and horizontal axis is again calculated with Equation (3). These values are again rounded to zero decimal places afterwards. The cell angles must now be compared to the actual gradient angles, for each passenger individually. This is done by generating a 5x5xn array, where each layer contains 25 times the gradient angle of one specific passenger centre cell. The absolute value of the difference between this array and *Degree* results in a 5x5xn array, where each layer contains the discrepancies of cell angles and the respective gradient angle. Now each cell with an absolute difference value greater than 22.5° is rated as unattractive and overwritten with the value 2. The centre cell in each layer is also overwritten with 2. All other values are normalized by 22.5°, so the cells that are rated as attractive have values between 0 and 1. Since only the absolute difference values have been considered, there are still movement directions rated as attractive which point in the opposing direction. For this reason, the

algebraic signs of all cell coordinates in each layer are compared with the algebraic signs of the x and y components of the respective gradient vector. In the case that these do not match, the movement direction is rated as unattractive and overwritten with the value 2. The attractive movement directions in each layer now have values between 0 and 1, where, consistent with *Direction*, the lowest is the most attractive value.

4.4.9. relocate_passengers

Input: *Attractiveness*, *R_mov*, *h*, *w,* pass_rmov*,* panic_level*,* v1*,* rmov_ind
Output: *PASSENGER_NEW*

This function relocates the passenger centres according to the values in *Attractiveness* and the present *panic_level*. *Attractiveness* is a 5x5xn array generated according to Equation (2). Each layer contains the resulting attractiveness values of all movement directions of one specific passenger. Firstly, all unreachable cells must be overwritten with NaN. For this reason, each cell that has a NaN value in *R_mov* is also overwritten with NaN in *Attractiveness*. Thanks to the same passenger order in *R_mov* and *Attractiveness*, this is easily implemented. Due to the influence of panic, there are two different movement options: Either the passenger decides rationally and chooses the most attractive cell, or he/she randomly decides for any of the reachable cells.

The rational decision is implemented by overwriting the lowest value in each layer with 1 and setting the other cells in the layer to 0. This 1 thus determines the new location of the passenger centre. The advantage of allocating unreachable cells with NaN is that MATLAB ignores NaN values in the process of finding the lowest value in each layer. The problem that another passenger could be created if the most attractive value in a layer occurs more than once is solved by correcting the affected layers: One of the value 1 cells is randomly chosen and the other cells are set to 0.

The arbitrary behaviour is implemented by manipulating the value of one randomly chosen cell per layer. It needs to be ensured that unreachable cells are not chosen, so the indices which lead to NaN cells are not considered. When one index per layer is chosen, the values in these cells are overwritten with -1. Now the lowest value of each layer is determined to be the new passenger centre, all other cells in the layer are set to 0. Setting the randomly chosen cells' values to -1 has a great advantage: It prevents the problem of generating new passengers, since all regular values in *Attractiveness* are greater or equal to 0.
The value of *panic_level* determines the probability for arbitrary behaviour. So n numbers between 1 and 100 are generated randomly, but with uniformly distributed probability, and stored in a vector. In the case *panic_level* $= 0.05$, for example, all numbers that are greater than 95 are overwritten with 0. All values that are lower or equal to 95 are overwritten with 1. The values in this vector now define the behaviour of the passenger. If the first value in this vector is 1, then the first passenger will behave rationally. If the third entry is 0, then the third passenger will behave arbitrarily and so on. This way the correct probability is realized.
A new 5x5xn array, called *ALL_val_total*, is created by assembling the specific layers of rational and arbitrary behaviour according to the probability vector. If, for example, the third entry in the probability vector is 0, then the third layer of the arbitrary behaviour is sorted in the third layer of *ALL_val_total*.

The final adaptation is to locate and correct passenger collisions. If two or more passenger centre cells decide for the same target cell, the index of this cell occurs more than once

in the new index vector of passenger centres. These indices are extracted and stored in a vector. The layers in *ALL_val_total*, that contain a collision index, are extracted. This is done consecutively for all collision indices. Of course it is possible that this index occurs in a layer, but the cell at this index does not have the value 1. So each layer that contains this index, but does not have a 1 in this cell, is not considered. Out of the remaining layers one is randomly chosen to keep the value 1 in this cell. The passenger centres in the other layers are relocated to their initial centre cell. Afterwards, the corrected layers are sorted in the correct position in *ALL_val_total*. Now the indices of the relocated passenger centres can be calculated. A new all zero sparse matrix with the dimension hxw is created. The cells at the new passenger centre indices are allocated with the value 1. This matrix contains the relocated passenger centres and is called *PASSENGER_NEW*.

5. Results

5.1. Image sequence of the simulation

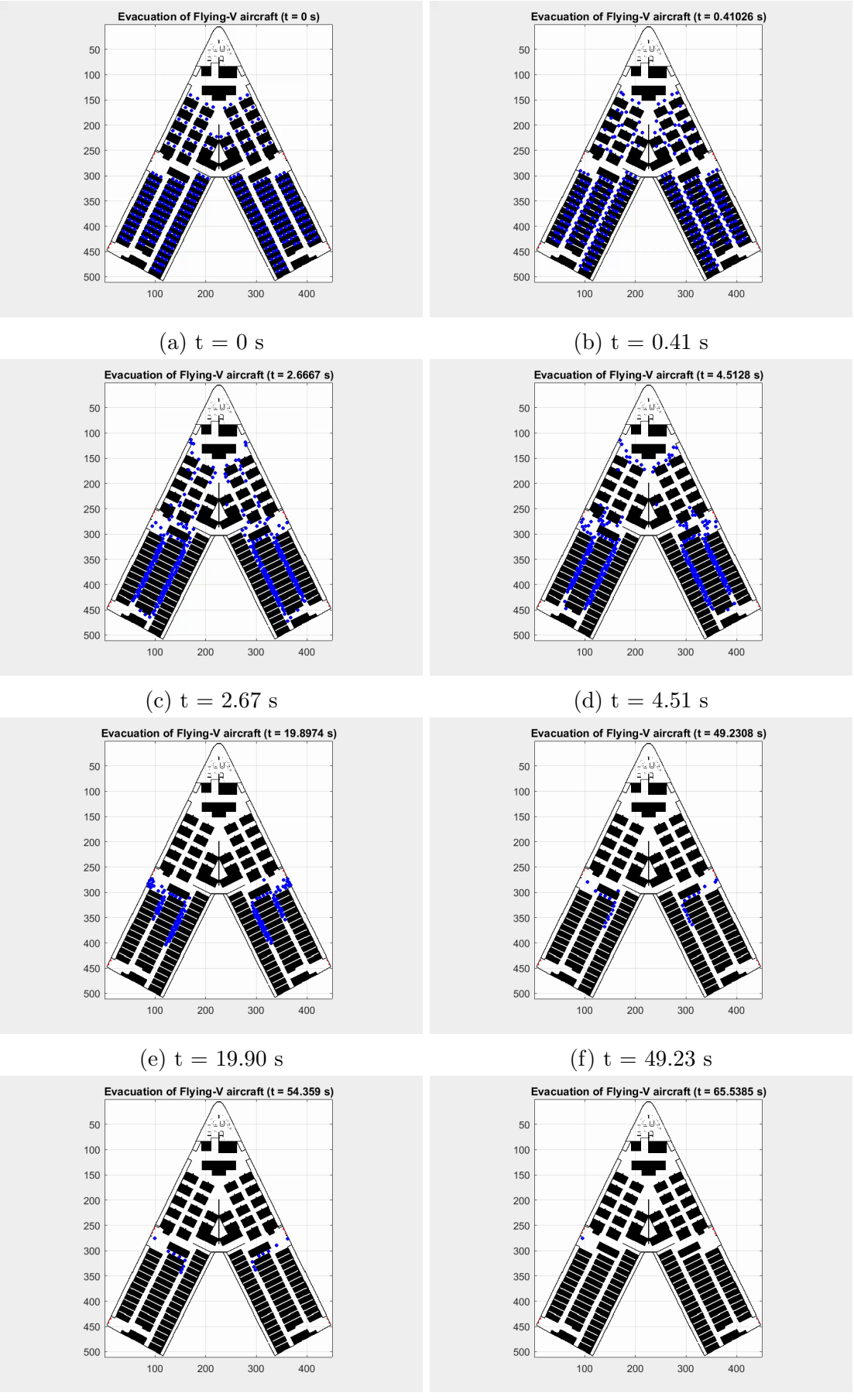
Due to the limited possibilities of illustration within this written thesis, only an image sequence of the evacuation simulation using the presented tool can be displayed. The videos belonging to each of the evacuation scenarios presented in this and the following sections as well as some additional cases can be accessed in the digital appendix. They are named after the section in which they are discussed. These videos will show the entire evacuation simulation for each case and will highlight all capabilities and limitations of this tool.

The image sequence in Figure 14 shows the evacuation simulation of the Flying-V aircraft using the presented CA tool. The corresponding video file in the digital appendix is, according to the section, called Section_5.1. The input and adjustment parameters were:

$$open\_doors = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, panic\_level = 0.05, k = 1, v\_max = 1.3m/s, t = 0.85, s = 0.8 \quad (4)$$

v_max is the velocity of 'fast' passengers and determines the time step size of the simulation. It is chosen according to [2] and [13], considering that 'slow' passengers only move with $v\_max/2$. t and s are the adjustment parameters to control the influence of *Distance*, *Gradient* and *Direction* according to Equation (2). Different values for t and s were tested and the ones mentioned above are considered to produce the best results. The impact of changing these values will be discussed in Section 5.3. The comparatively low panic level of 5% seemed appropriate for simulating a certification test since all participants in those tests are aware that there is no actual danger. In order to simulate a real evacuation scenario, it is expected to be higher. The impact on the passengers' behaviour by changing this specific value will be discussed in Section 5.4. The closed doors are represented as red squares in the plot. The images in Figure 14 were chosen in order to show different stages throughout the evacuation process. Thus, they are not equally distant in time. The time values in the captions are the rounded values of the exact time that can be seen in the title of the plots.

As mentioned in Section 4.1, special attention must be paid to the passengers seated in the first rows of Economy and Business Class. If the doors in front of these passengers are open, they do not have to reach the main aisle first, but evacuate immediately. In the initial geometry file shown in Figure 6 the area around these passengers is not considered as aisle area in order to allow this behaviour. This is the reason why the *open_doors* vector has been chosen the way it is for the image sequence: This specific door constellation ensures that there is an open door in front of each of the affected rows. In the case that the doors in front of the affected passengers are not open, the geometry file needs to be adapted. This will be discussed in further detail in Section 6.2.



(a) t = 0 s (b) t = 0.41 s

(c) t = 2.67 s (d) t = 4.51 s

(e) t = 19.90 s (f) t = 49.23 s

(g) t = 54.36 s (h) t = 65.54 s

Figure 14: Simulation of Flying-V evacuation

The image sequence shows that all passengers leave the aircraft, the last one after approximately $65.54s$. At the start of the simulation process, all passengers are located in front of their seat. Figure 14b accurately shows how the passengers leave their seat aisles to reach the main aisles. As expected, the passengers sitting in the central seat rows leave the seat aisle to both sides. The passengers seated in the front rows of Economy and Business Class directly approach the doors without reaching the main aisle first. In Figure 14c, it can be seen that a jam of passengers arises in the main aisles after all passengers have left the seat aisles. Due to the different number of passengers, the Business Class is evacuated except for one passenger when there are still many people in the Economy Class, which can be seen in Figure 14e. Clearly visible in the video file is also the difference in velocities: If the passengers leave the proximity to a wall or to other passengers, they instantly accelerate. The total evacuation time of approximately $65.54s$ implies that the 90-second-limit is clearly undercut. However, due to the limited influencing factors for passenger behaviour, as well as the negligence of phenomena like seat jumping, and the low panic level this value should only be seen as a first approach in measuring the time. All limitations and problems in this preliminary tool will be extensively discussed in Section 6.

5.2. Influence of PNG file resolution

The accuracy of the simulation depends on the resolution of the PNG file that is imported. Even with the lowest accuracy ($k = 1$), the passengers' movements seem relatively natural. For $k = 1$, the program runs very quickly, although the matrices are already comparably big. One of the reasons for this is that aside from the time loop the code contains very few other loops. With increasing accuracy, the passengers' movements become less angular, but evidently the program becomes slower. Additional difficulties in the passenger behaviour arise which will be discussed in Section 6.

5.3. Influence of adjustment parameters t and s

The adjustment parameters t and s (both between 0 and 1) control the influences of the different fields for passenger decision. The equation for the resulting attractiveness values is again:

$$Attractiveness = t \cdot (s \cdot Distance + (1 - s) \cdot Gradient) + (1 - t) \cdot Direction$$

s thus controls the weighting of distance and gradient values. Both are counted in the following to 'distance-related factors'. t controls the weighting of the distance-related factors and the *Direction* field. As explained, *Direction* considers the position in the seat aisles and the proximity to obstacles and passengers. Without this field, most of the passengers would either not leave the seat aisles or even enter seat aisles again when they have already been on a main aisle. Additionally, *Direction* prevents the passengers from getting too close to obstacles and other passengers.

Increasing t makes the distance-related factors more important and the *Direction* values less important for the passenger decision. This results in a very determined behaviour of the passengers towards the exits. The values in *AISLE_POSITION* are, even for a high t value, high enough to force the passengers out of the seat aisles first. There is, however, a limit in increasing t: The higher t, the closer the passengers get to walls and other passengers. So it seems that they all coalesce to one moving mass which must be

prevented. A low value for t leads to a directionless behaviour and causes problems at the exit cells: A high influence of *Direction* forces the passengers to avoid any contact with obstacles. And since there are still obstacle cells around the exit cells in the geometry plot, the passengers cannot or only with difficulty reach the exit cells. By testing the optimal value was found out to be $t = 0.85$.

Increasing s raises the influence of the distance field and reduces the influence of the gradient field. A high s value has the following characteristics: The passengers are able to move around the corners, although the gradient vector might point directly through the obstacle. A disadvantage, however, is that on their way to the next open door, the passenger often glide along walls. They want to get close to the door side quickly and then approach the exit cell by following the wall. This results in a lower passenger velocity. Another disadvantage is that some passengers move slightly zigzag-shaped in the case that their way is diagonal. A low s value results in a more natural behaviour: The passengers try to follow the straight line of the gradient vector if possible. Additionally, a higher gradient influence makes the decision of passengers that are far away from the next open door more distinct. The limiting factor is, however, the movement around the corners: As mentioned, situations in which the gradient vector points directly through an obstacle are problematic. With a low s value, passengers get caught in these situations and cannot leave the aircraft. Evacuating the entire number of passengers needs to be ensured, so this must definitely be prevented. The value fulfilling these requirements best was found to be $s = 0.8$. The corresponding videos can be found in the digital appendix.

5.4. Influence of v_max and panic_level

In this section, the impact of modifying *v_max* and *panic_level* on the passenger behaviour is analyzed. As previously mentioned, *v_max* determines the time step size of the simulation. The underlying formula is:

$$\Delta t = \frac{2 \cdot l \cdot 10^{-2}}{v\_max} \tag{5}$$

l is the length of one field cell in cm for the desired accuracy. Since 'fast' passengers can overcome two cells in one time step, the maximum distance covered is $2l$. The difference in distance values between horizontal/vertical and diagonal cells in the movement radius is neglected. For the analysis, the number of passengers in the aircraft is plotted over the time in seconds. The door constellation, the accuracy, and the adjustment parameters are the same as in Section 5.1. For the panic level, a high and a low value were tested: $panic\_level = 0.65$ and $panic\_level = 0.05$. The high value is regarded as an upper limit, since it means that over 50% of the passengers' decisions are random. Even for a real emergency evacuation, this value might be doubtfully high, but will make the impact of *panic_level* evident. Additionally, three different values for the maximum velocity were tested: $v\_max = 0.6m/s$, $v\_max = 1.3m/s$ and $v\_max = 2m/s$. The maximum velocity affects the total evacuation time as it determines the time step size, but does not influence the passengers' movements. In this tool, the movement radius always is a 5x5 block and the velocity is taken into account with Equation (5). The velocity does not change the movement radius. Therefore, an increasing velocity always leads to faster evacuation. However, this is not realistic: Up to a certain point, higher velocity values should result in faster evacuation. Above this point a further increase should lead to more collisions

and therefore slower evacuation. According to [2], "[...] desired velocities higher than about 1.3m/s reduce the efficiency of leaving." This effect cannot be seen in Figure 15 due to the underlying idea of the movement radius.

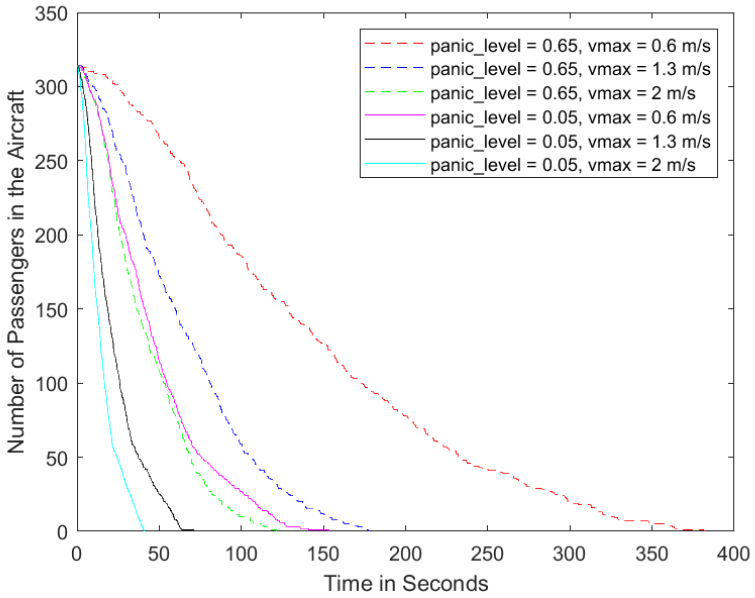


Figure 15: Impact analysis of *v_max* and *panic_level*

In all of the cases in Figure 15, the expected course is visible: The negative slope of the graphs increases up to a maximum value and decreases again until the graphs almost horizontally approach the x axis. This seems realistic: In the beginning, all passengers need to leave their seats first and start approaching the doors. Only the ones who were seated very close to the doors can reach the exit cells at this stage. Then, jams arise at the doors and the open ones are used to full capacity. At the last stage, most of the passengers have already left the aircraft, the remaining passengers are the ones who were seated far away from the next open door. Remarkable is that even with a panic level of 65% all passengers are able to leave the aircraft. The graphs show that both *panic_level* and *v_max* have a significant impact on the evacuation time: A lower panic level leads to a substantially faster evacuation. A higher maximum velocity also has the same result. The proportion of the different velocity graphs matches with the ratio of the different values of *v_max* if the ratio of the velocities and the ratio of the respective total evacuation time are compared. By considering the total evacuation time and observing the passengers' behaviour during the evacuation process, a panic level of 5% still seems appropriate for simulating a certification trial. In order to determine a specific panic level for a real emergency evacuation, further studies would be necessary. The maximum velocity of $v\_max = 1.3m/s$ is used so that it corresponds with the results in [13] and [2]. For both panic levels, a video file of the evacuation simulation can be found in the digital appendix.

6. Discussion

In this section difficulties and problems of the presented tool will be discussed to motivate further improvement.

6.1. Colour values in the PNG file

As explained in Section 4.2, the PNG file is automatically analyzed by MATLAB to generate the geometry matrices. In this process, the PNG file is transferred into a matrix where each cell contains the value of the colour in this specific pixel. These values correspond to the position of the colour on the colour palette. Unfortunately, the values are not the same for different accuracies: Although, for example, the aisle areas for $k = 1$ and $k = 2$ were coloured with exactly the same shade of red in the Paint.net file, these pixels' values differ in the matrix. This problem is easily solved by looking at the values in the matrix after MATLAB imported the PNG file. By comparing the matrix with the underlying Paint.net file, it becomes obvious which value belongs to which colour. A reason for this problem might be the total number of different colours in the file: The higher the pixel density, the more shades of a colour are used for the colour transitions. It would be more convenient, if the same colours always resulted in the same colour values, regardless of the accuracy.

6.2. Passenger behaviour

The door constellation for the results presented in Section 5 was chosen for a trouble-free simulation with the initial PNG file. As previously mentioned, difficulties occur when the doors in front of the passengers in the first rows of Business and Economy Class are not open. The passengers in these rows cannot find the way to the open doors and stay in front of their seats (visible in the video file Section_6.2_Problem). In such a case, these passengers also need to be forced to reach the main aisles. From there, the way to the next open door is clear. This problem was already addressed in the course of this work by introducing an adapted PNG file of the geometry: In the adapted file, the area around the passengers in the first rows was also coloured in red in order to highlight it as additional aisle area. Therefore, the *AISLE_POSITION* field was also applied to these cells. This ensures that the passengers in the first rows are forced to reach the main aisle. The passenger centres in the affected rows also needed to be placed a little bit closer to the main aisles. For a trouble-free passing by at the sanitary facilities in the Business Class some additional aisle areas have been defined. With these adaptations, all possible door constellations can be successfully simulated. For each accuracy, an adapted PNG file has been created that can be chosen in the MATLAB function *get_geometry*. With the adapted PNG files, the initial case can also be successfully simulated, but the behaviour of the passengers in the first seat rows seems unnatural: They enter the main aisle first before approaching the doors instead of walking straightforward. Therefore, in the current state, it is advisable to choose the PNG file in *get_geometry* according to the desired door constellation. The only door constellation that is still problematic is the case of all four doors of the same side being open. In the case that each door on the left side is open, all passengers from the right side need to move there. This is not a problem for the passengers in the Business Class, but for the passengers in the Economy Class: There is a support wall between Business and Economy Class visible, for example, in Figure 6. The passengers need to move around it to reach the other side of the aircraft. It is possible to

simulate this case with the presented tool, but the area below this support wall also needs to be classified as aisle area. This forces the passengers to move around the support wall. This adaptation has only been applied to the accuracy $k = 1$ since it is a very special case. Hence, for $k = 1$ there are three, for $k = 2$ and $k = 3$ two different PNG files to choose from in *get_geometry*. For the cases explained in this section, the videos of the simulation process can be found in the digital appendix.

Regardless of the accuracy, individual passengers in the Economy Class are sometimes not able to leave the seat aisle and stay in front or their seat. With increasing accuracy, this phenomenon happens more often and when it occurs, the simulation needs to be aborted and restarted. The reason for this probably lies in a random movement since it only happens occasionally and for various passengers. This random movement combined with an inaccuracy of the *AISLE_POSITION* field might cause the problem that the passenger is not able to leave a particular cell again. For the high panic level in Section 5.4 it could, however, be observed that in such a situation, the passenger still left the seat aisle after a while. Another random movement can possibly free the passenger from this situation. The same problem occurs in the Business Class: Passengers get caught on the armrests between the seats. For $k = 1$, a random movement most of the time frees the passengers and enables them to leave the seat aisle. For higher accuracies, these passengers are often not able to leave the seat aisle and the simulation needs to be restarted. This problem could be solved by improving the *AISLE_POSITION* field: Getting caught on a seat aisle cell must definitely be prevented.

Another behaviour that seems unnatural is when there is a jam in the main aisle: Although the other main aisle might be empty in this half of the airplane, none of the passengers change to the other main aisle, but all of them wait until the jam in front of them is cleared. This is not a mistake in the code, but clearly a limit of the underlying concept. The seat aisle cells have very unattractive values to force the passengers into the main aisles and to prevent them from walking into a dead end. Crossing the seat aisles of the middle seat rows, however, does not lead into a dead end. Another possibility for the passengers would be to aim for another open door which may be further away, but is not jammed. According to the concept of the tool, this is not possible since the waiting duration at the doors does not influence the movement decisions. Both ideas could lead to an improvement in the passenger behaviour and will be reexamined in Section 7.

One particularly difficult situation is depicted in Figure 16. The passengers in the right main aisle (in the picture) need to get around the corner of the first middle seat row to reach the door. Moving around this corner is not intuitive for the passengers since they have to step on cells that are less attractive in terms of *Distance*. The gradient influence and in some cases an advantageous random movement, however, render this manoeuvre possible. Other passengers behind the one currently trying to get around this corner also facilitate this movement: Moving forward, they practically push him/her around the corner. In seldom cases, the last passenger is not able to move around the corner or at least takes more time than expected. By allocating these cells at the corner with special values, this difficulty could be solved.

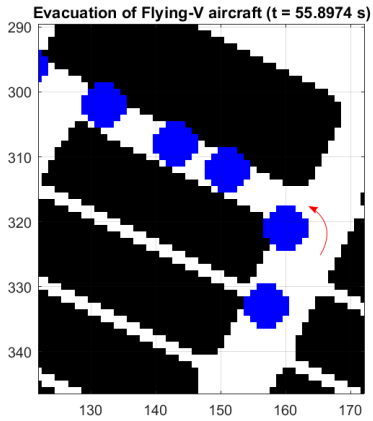


Figure 16: 'Corner manoeuvre'

6.3. Simulation accuracies

For $k = 1$, the simulation runs very quickly and, as explained in the last section, mostly trouble-free. The resolution is accurate enough to show relatively natural behaviour of the passengers. With increasing accuracy, a problem at the exit cells occurs which is reasoned in the way the overlapping passenger radii are handled: The overlapping cells are counted. With higher accuracy, the number of cells in each passenger radius increases meaning that the area of overlapping passenger radii contains a greater number of cells for higher accuracies. Getting close to each other is therefore less attractive for a higher k value. This is clearly identifiable by the greater jams at the doors for higher accuracies. A solution could be to normalize the number of overlapping cells by the total number of cells in each passenger radius. Since the improvement in the naturalness of the passenger behaviour does not compensate the arising problems and the higher computational costs in the current state, a higher accuracy than $k = 1$ is not recommended.

6.4. Radius of movement

In this tool, the radius of movement is always a 5x5 block. This allows very efficient coding, but leads to some difficulties: Firstly, there are only two different velocities. Only the fact that there is at least one overlapping cell in the passenger radius determines that this passenger moves slowly. The actual number of overlapping cells does not matter. This could be improved by extending the radius of movement: It would be possible to introduce more than two different velocities. Thus, the number of overlapping cells could influence the passenger's velocity.

In the current state, the diagonal cells in the movement radius can be reached in the same time as the horizontal or vertical cells. This means that the distance difference of the diagonal cells is neglected. For a more accurate simulation the fact that the diagonal cells are further away than the horizontal and vertical cells should be considered.

7. Conclusion and Future Work

In this thesis, a preliminary CA tool for an emergency evacuation simulation for the Flying-V aircraft has been presented. A major aspect of this work was to efficiently implement the airplane geometry. This was accomplished by generating an adapted PNG file of the interior and transferring it to MATLAB. This allows an efficient and accurate extraction of the information on the different areas in the passenger cabin. This method is generally applicable to any other airplane geometry. The passengers in this tool are represented by a centre cell and a uniform, the passenger surrounding radius. This makes two different movement velocities possible that are clearly visible in the simulation process. The accuracy, the open doors and the panic level of the passengers can be defined by the user. Each developed MATLAB function used in this tool has been explained in detail to facilitate further improvements.

The results have been presented and critically analyzed. A variety of simulation videos can be found in the digital appendix attached to this thesis. This tool is capable of successfully simulating each possible door constellation with relatively natural passenger behaviour, but further improvements are necessary for a reliable and more accurate evacuation simulation.

The inconvenient situation of choosing a PNG file according to the entered door constellation needs to be addressed: This means that either a MATLAB function automatically decides which PNG file to choose according to the *open_doors* vector or, even better, special influence fields are created for the passengers in the front rows and the area around the support wall. In order to find the affected cells of these areas, the PNG file could be adapted with an additional colour. To solve the difficulty of the 'corner manoeuvre', the cells' values also need to be adapted in the affected areas.

The *AISLE_POSITION* field also needs further improvement: Inaccuracies which might result in passengers sometimes not leaving their seat aisle need to be corrected. For the middle seat rows, a temporary *AISLE_POSITION* field that is deactivated once these passengers left the seat aisle could be advantageous: Thus, the passengers could switch main aisles in case that one is more jammed than the other. It must, however, be ensured that not all of them switch to the other side. This could be realized by another influence field considering the number of passengers between the aimed door and the regarded passenger. By doing so, an approximate waiting duration could be calculated and compared to other possibilities for this passenger. If another open door is further away but less jammed, this field could also influence the passenger's decision to aim for the other door.

An additional aspect for a more realistic simulation could be to introduce different passenger radii with appropriate movement velocities. This would make it possible to represent the specific passenger load required for the certification test. The presence and influence of a cabin crew have not been considered yet. It would be interesting to implement a decreasing panic level of the passengers the closer they are to crew members. These crew members should leave the aircraft only once all passengers are evacuated. Group dynamics like families or partners staying together during the evacuation process could also be considered. In order to conveniently enter the input parameters, a Graphical User Interface (GUI) could be created.

The possibilities of additional aspects for the simulation are endless. In the current state, this tool is only a basis. The results presented in this thesis, however, revealed that with further improvements a fast and accurate tool for the evacuation simulation of the Flying-V aircraft can be developed.

References

[1] European Union Aviation Safety Agency. Certification specifications and acceptable means of compliance for large aeroplanes cs-25. January 2020.

[2] R. Alizadeh. A dynamic cellular automaton model for evacuation process with obstacles. *Elsevier Ltd.*, 2010.

[3] International Air Transport Association. New iata passenger forecast reveales fast-growing markets of the future, 2014. https://www.iata.org/en/pressroom/pr/2014-10-16-01/, Last accessed 15 May 2020.

[4] Justus Benad. The flying v - a new aircraft configuration for commercial passenger transport. *Deutscher Luft- und Raumfahrtkongress, Rostock*, September 2015.

[5] Hong bing Du; Xiao-fang Yu; Xin Fu; Zhen-yu Feng. Research on occupant evacuation simulation during civil aircraft emergency. *Elsevier Ltd.*, 2018.

[6] Air Accident Investigation Branch. Report on the accident to boeing 737-236 series 1, g-bgjl at manchester internationl airport on 22 august 1985. Technical report, December 1988.

[7] European Transport Safety Council. Increasing the survival rate in aircraft accidents. December 1996.

[8] TU Delft. Flying-v. https://www.tudelft.nl/en/ae/flying-v/, Last accessed 24 May 2020.

[9] TU Delft. Flying-v. https://www.tudelft.nl/en/ae/flying-v/technology/, Last accessed 21 July 2020.

[10] Minesh Poudel; Bhaskar Chaudhury; Kshitij Sharma; Pavel Yaroslavovich Tabakov; Félix Mora-Camino. Gpu based computational simulation of aircraft evacuation: Temporal and spatial analysis. *Elsevier B.V.*, 2018.

[11] U.S. Office of Technology Assessment. Aircraft evacuation testing: Research and technology issues. September 1993.

[12] E.R.Galea; S.J.Blake; P.J.Lawrence. The airexodus evacuation model and its application to aircraft safety. *Researchgate*, January 2002.

[13] Yugang Zhang; Zhaohui Yang; Zhongchao Sun. A dynamic estimation method for aircraft emergency evacuation based on cellular automata. *Advances in Mechanical Engineering*, 2019.

[14] Tomas van Dijk. Delft researchers are building an aircraft without tail, June 2019. https://www.delta.tudelft.nl/article/delft-researchers-are-building-aircraft-without-tail, Last accessed 24 May 2020.

[15] Stephen Wolfram. Universality and complexity in cellular automata. *Physica*, 1984.

A. Appendix

A.1. Adaptation of the Paint.net file

After cutting out the passenger cabin out of the interior concept picture (see Figure 4a) and transferring it to Paint.net, a pencil sketch is created. In this step the tinge is set to zero (a black and white picture shall be created) and the contour value is set to maximum. Now the manual adaptations start: Firstly, it needs to be ensured that there is no gap in the outer walls. This is very important since already one missing wall pixel can enable passengers to leave the aircraft through this gap. All pixels that are walkable area need to be white. All obstacle cells need to be black. Hence, all pixels which belong to seats, sanitary facilities and other obstacles need to be coloured black. The contours of the obstacles should be as uniform as possible to reduce the risk of passengers getting caught on corners and edges. Cells with transition colour shades can be disregarded: If these cells are not important for the geometry information they do not need to be deleted. Since they do not have a colour shade that is considered in the program, they are simply neglected while analyzing the PNG file.

The inner wall at the doors must be erased, so that the passengers are able to reach the outer wall, where the exit cells are going to be located. Now the seat aisle areas are coloured red. For all colours the standard shade, that is suggested in the colour palette in the lower right corner of the program, has been used. This is, however, not obligatory as long as the value of the chosen colour can be clearly identifiable in MATLAB. Two exit cells per door are placed: Therefore, two cells in an optically appropriate distance are coloured yellow at each door. The passenger centre cells are placed by colouring one cell per passenger blue, centered in front of the corresponding seat. The passenger centres in the first rows of Business and Economy Class are coloured green. Now this image can be saved as a PNG file with a bit depth of 8 bit. The Dithering Level is set to zero in order to prevent colour transitions.

For the 'adapted PNG file' (see Section 6.2), the area around the passengers in the first rows of Business and Economy Class also need to be coloured red. The shape of the seat aisle area in these rows needs to be created by testing: The affected passengers need to reach the main aisle, but the impact on the other passengers while passing this area must be as low as possible. The adaptation requires that these passenger centres are coloured blue instead of green. They are also placed slightly closer to the main aisle to facilitate the movement. The first middle seat row in the Economy Class does not have to be adapted: The simulation showed that these passengers behave correctly for all door constellations. For a trouble-free passing by at the sanitary facilities in the Business Class, additional aisle areas were defined. All corresponding Paint.net files for the PNG images used in the presented tool can be found in the digital appendix. This might be helpful for a better understanding and further adaptations.