Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Pedestrian Dynamics Airplane Evacuation Simulation

Philipp Heer & Lukas Bühler

Zurich
May 2011

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Philipp Heer　　　　Lukas Bühler

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of Originality

This sheet must be signed and enclosed with every piece of written work submitted at ETH.

I hereby declare that the written work I have submitted entitled

____

is original work which I alone have authored and which is written in my own words.*

Author(s)

Last name	First name

Supervising lecturer

Last name	First name

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (http://www.ethz.ch/students/exams/plagiarism_s_en.pdf). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature

*Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Print form

Contents

1 Individual contributions

The following Tasks were performed:

Task	**Responsibility**
Converting image files to matrices (file getFile.m)	Lukas Bühler
Redrawing of seating plans	Lukas Bühler
Generation of the vector field containing the shortest path to an emergency exit (file computeGradientField1.m)	Lukas Bühler
Analysis of the simulation results with different positions of the flight attendants	Lukas Bühler
Calculation of acting forces on passengers (file pedest.m)	Philipp Heer
Programming of Simulation-Algorithm (file pedest.m)	Philipp Heer
Converting matrices from getFile.m into a usable form (file init.m)	Philipp Heer
Analysis of the simulation results with parameters	Philipp Heer

2 Introduction and Motivations

Figure 1: Evacuation of an Airbus A320 after the successful emergency landing in the Hudson River, Source: (3)

In January 2009 an Airbus was forced to land in the Hudson River in New York. At this event 150 persons had to leave the plane before it sank to the ground of the river. In this project the evacuation of airplanes is analyzed. After a crash it is possible for an airplane to explode due to ignited fuel or after a water landing to sink. During such an emergency situation it is important to evacuate all passengers and crew members of an airplane in a very short amount of time in order to maintain their survival. Evacuation scenarios were studied before. Emerging fires or earthquakes lead to similar situations in buildings. Even the evacuation of airplanes has already been researched because hundreds of lives could be at stake. The International Civil Aviation Organization (ICAO) limited the maximum time of evacuation to 90 seconds for each plane. This time reference has to be proved by each manufacturer before a plane can go into service.

Especially the newer airplaes have even higher decks than older ones. This raises the question whether people fear to jump down the slides in an emergency situation.

Is there a psychological aspect that is not answered by normal pedestrian dynamics adapted on the space of an airplane in an emergency situation? This question led to the fundamental questions for this Project:

2.1 Fundamental Questions:

Is this model suitable for the evacuation situation of an airplane or should there be done changes in order to get reliable results? Example: Do people hesitate to jump down the slides at the exits and does the social force model fail there?

Could suitable instructions of the cabin personal be simulated in the model and how does it affect the performance of the scenario?

2.2 Expected Results

There is data of evacuation experiments with real aircrafts available. This will help to verify and test the model. The answer of the question will be the confirmation of the test results by the model or a succession to improve the model. If there are differences of our model to the original social force model, they should be pointed out and explained.

We expect the instructions of the personal to be a positive effect on performance. To simulate this, there will be changes in the original social force model.

2.3 Research Methods

Continuous Modeling by implementing the social force model.
The model time step dt will be defined. The performance of the model will be measured by counting the time steps. By counting the amount of time steps, the model is validated relatively and can not be compared to a real time value. Nevertheless, this allows us to compare adaptations of our model.

3 Description of the Model

It was the aim to implement an easy adaptable model of an aircraft in which the simulations of pedestrian dynamics in an emergency situation could be performed. Therefore, a slightly reworked seat plan image of the airplane can be imported into Matlab to generate the environment of the simulation. Walls, space, passengers, flight attendants, emergency exits and special zones are drawn with different colors in this image. Matlab first generates the shortest path from every point to an exit and generates therefore a vector field in the space of the airplane. Every passenger basically follows this path to the next emergency exit from his position.

The main simulation is then performed with a implementation of the social force model described in (1).

During a simulation passengers interact with each other and can therefore not follow the shortest path given by the vector field. These interacting forces consist of basical phisical principles (e.g. that passengers cannot walk through walls or that two passengers cannot stand too close to each other) and social interactions which are introduces because generally persons try to have a free region of space around them to feel comfortable.

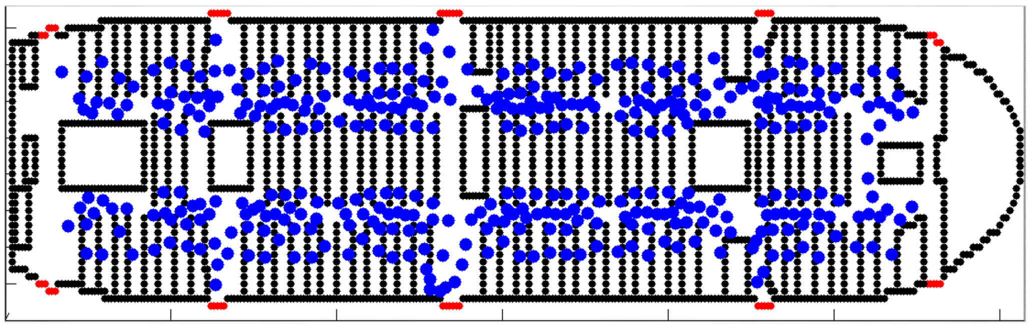


Figure 2: Outputframe of a simulation

4 Implementation

The whole program is gouped into two major parts as shown in Figure 3. First the init.m file is executed in which the airplane picture can be choosen and the path finding algorithm runs. Also each a matrice for the passengers, the flightattendants and the wallobjects is generated. Secondly, the pedest.m file computes the time iteration and generates the output.

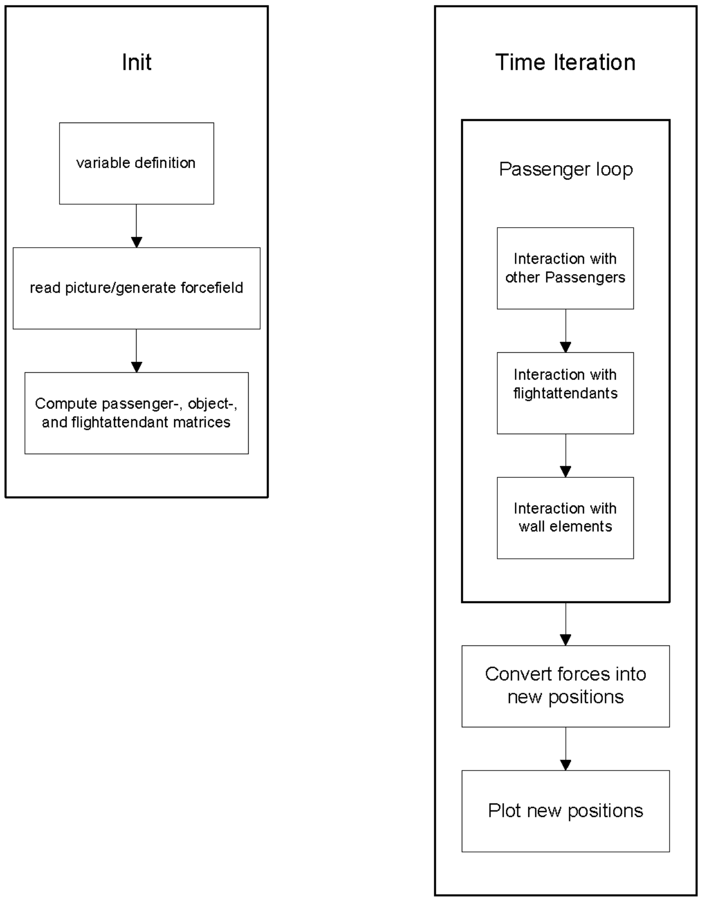

Init
variable definition
read picture/generate forcefield
Compute passenger-, object-, and flightattendant matrices
Time Iteration
Passenger loop
Interaction with other Passengers
Interaction with flightattendants
Interaction with wall elements
Convert forces into new positions
Plot new positions

Figure 3: Flow chart of the initialization file (init.m) and the time iteration file (pedest.m)

4.1 Initialisation

On initialization, the space for the simulation containing walls, exits, passengers, special areas and the vector field with the shortest path is generated. These processes are separated in different matlab files. First, an image file is loaded and processed by the file getFile.m. It generates a matrix. In the computeGradientField1.m file, this matrix is searched for walls and exits and the shortest path from every point to the next exit is generated in a vector field. The passengers in the following simulation basically follow this field.

4.1.1 Importing an airplane Model

There are many so called seating plans of aircrafts available on the internet. They are mostly provided by airlines for their customers in order to choose a seat while booking. Those images are used to generate a realistic model of the environment in a short time in order to perform many simulations with different shapes and conditions of the environment. Adobe Fireworks was used to rework the seating plan. Figure 4 shows a seating plan and the reworked image to import into Matlab. Matlab offers functions to convert an image file into a matrix with every entry representing a pixel of the image.

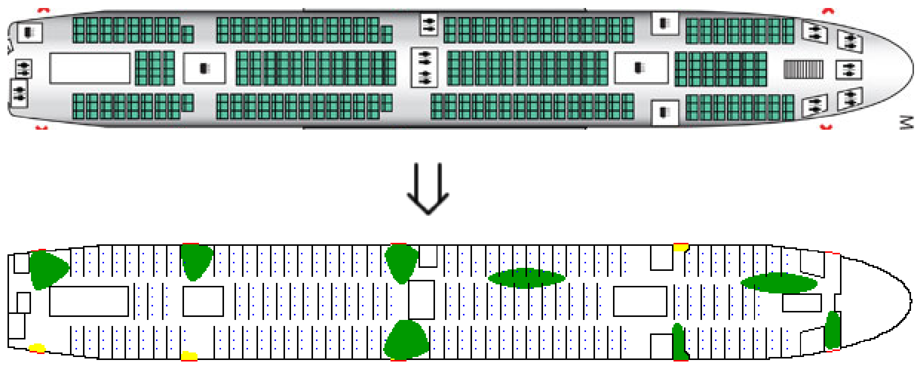


Figure 4: Airplane seating plan and reworked image file to import into Matlab

The image file (bitmap) must contain the in Figure 5 listed colors in order to recognise the different regions.

In the getFile.m file, every blue pixel is recognised als a passenger at its start position. The dark green zone is influenced positively by a flight attendant. In yellow areas,

Color	Hex triplet	Description
White	FFFFFF	Space in which the passengers can freely walk
Black	000000	Walls
Red	FF0000	Emergency exit (drain which attracts the passengers)
Blue	0000FF	Every blue pixel is recognised as a passenger
Dark green	009900	Zone which is influenced by the flight attendant
Yellow	FFFF00	Area in which pessangers struggle to continue walking

Figure 5: Colors of the image file and their meaning

passengers walk slower because they struggle to jump down the emergency slides. The output of getFile.m is a matrix which contains different numbers for walls, space, exits, passengers and zones.

4.1.2 Path finding algorithm

Every passenger must be guided along the shortest way to the next emergency exit. A vector field has to be found to get the direction to the next exit at every point of the space. Therefore, a fast marching algorithm is used which calculates the distance between every point and the next exit around every obstacle. Hence the problem of finding the shortest way between two points in space is similar to wave propagation problems or spreading fluid, there could be found a suitable implementation in the internet. A toolbox was found from Mathworks.
`http://www.mathworks.com/matlabcentral/fileexchange/6110`
The fast marching algorithm generates a potential field in the space, which contains the distance of every point to the next drain (emergency exit).

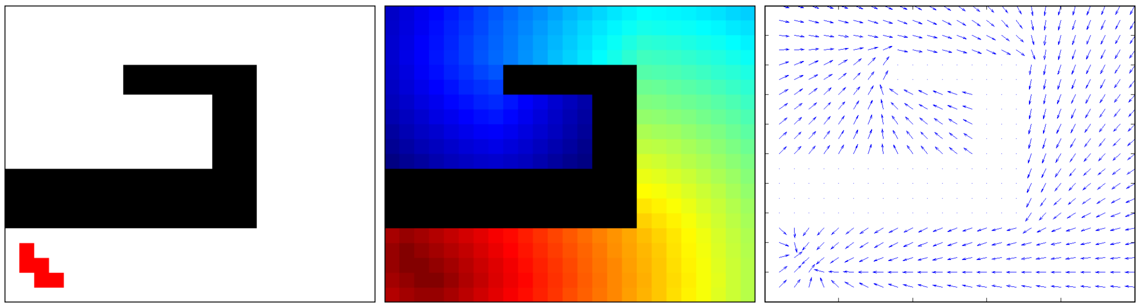


Figure 6: Developement of the vector field containing the shortest path to the next emergency exit

Figure 6 shows the generation of the vector field with the shortest path to an exit. The first image is the image file imported into Matlab with space (white), drain (emergency exit, red) and walls (black). With the fast marching algorithm a potential field is generated. Red represents the closest region to the exit and blue is the furthest. Via the calculation of the gradient of this potential field, the vector field in the right image is generated. The arrows point in the direction of the shortest path from every point to the emergency exit.

For the calculation of the gradient field, the matlab function gradient_special.m was written. The standard Matlab function gradient(X) could not have been used because of the exeptions it had to consider for walls. The function gradient(X) considers the whole matrix and calculates wrong values next to walls, because the distance entry of a wall is 0 in the matrix.
At the end of the calculation, all vectors are normalized to a constant length of 1 because only their direction is being used.

Figure 7 shows the fast marching algorithm applied on an imported aircraft model. It is assumed that the way to the emergency exit in an aircraft is marked so that a passenger always takes the nearest exit from his seat. With this assumption, the fast marching algorithm suits for generating the vector field in an airplane.

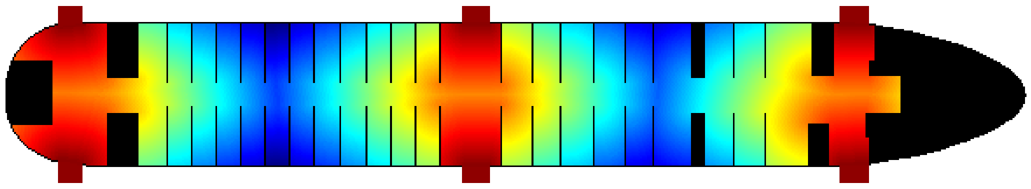


Figure 7: Visualized matrix that contains the distance to the next emergency exit in a Boeing 737

4.2 Running Simulation

After the initialisation the simulation beginns to run until every passenger has escaped the airplane.Each iteration step can be devided into three groups.
First - for each passenger - its actual position is analized. If it is in a zone which is influenced by a flight attendant it will behave differently from being in an area close to an exit. If the passenger is in an exit area it will be taken out of the simulation and is "free".
Secondly, the new position is calculated. For that wall elements and other passengers close to it self are looked at.

4.2.1 Generation of Interaction forces

These forces are divided into three parts. The first force to introduce is the so called physical force. It allows each passenger to have an area of space in which no one else can enter. It states for the body volume of a passenger. It is implemented as exponential function which decays fast and only is taken into consideration if two passengers are situated closer than one meter from each other.
Secondly, a social force is introduced which is over all weaker than the physical force but it decays slower and has a radius of action of six meters. Furthermore it is not isotrop. With an increasing variable λ the passanger reacts stonger to intrusions in front of it than behind of it.
Basically both the physical and the social force base on the principle the force increases the closer two passengers are together. This is achieved by using an exponential approach:
For the physical force between passenger a and b applied to a

$$\vec{F}^{phy}_{a-b} = A^{phy}_a \cdot e^{\frac{r_a + r_b - d_{a-b}}{B^{phy}_a}} \cdot \vec{n}_{a-b}$$

is applied and for the social force

$$\vec{F}^{soc}_{a-b} = (\lambda + (1-\lambda)\frac{1+\cos\phi_{a-b}}{2}) \cdot A^{soc}_a \cdot e^{\frac{r_a + r_b - d_{a-b}}{B^{soc}_a}} \cdot \vec{n}_{a-b}$$

holds true, where $\vec{n}_{a-b} = \frac{\vec{r}_a - \vec{r}_b}{||\vec{r}_a - \vec{r}_b||}$. ϕ_{a-b} is the angle between $\vec{n}_{a-b}$ and the direction of movment of a. A_a is the interaction strength and B_a the range of the interaction force. Both A_a and B_a depend on a as these parameters depend on the passenger e.g. on the size age, cultural background and so forth.
Figure 8 shows the equiforcelines of a passengers social force located at (0,0) headed in y-direction with a $\lambda < 1$.
As a third essential element a force has to be introduced which unables passengers to walk through walls. Whereas the forcefield generated by the pathfinder does not

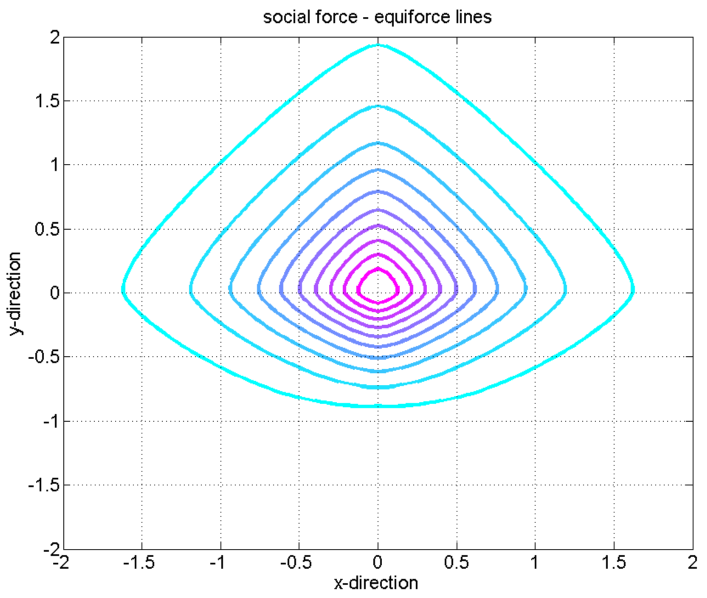


Figure 8: Unisotropic social force

point towards an obsticle as a wall but the other two mentioned forces could force a passenger rigth through a wall. For that reason each passenger too close to a wall element is pushed in normal direction to the orientation of the wall.
In a next step the new position of each passenger is calculated with Eulers Method applied on Newtons Laws of Motion.

$$\vec{F}_{total} = \sum_i \vec{F}_i = m \cdot \vec{a} = m \cdot \vec{\ddot{r}} \approx m \cdot \frac{\dot{r}(kT) - \dot{r}(T(k-1))}{dt}$$

with $\dot{r}(kT) = \frac{r(kT)-r(T(k-1))}{dt}$; $r(kT) = \begin{pmatrix} x(kT) \\ y(kT) \end{pmatrix}$; $k \in \mathcal{N}$ and $T = dt$
After that the new positions are plotted and - if there are still passengers fleeing - a next iteration is executed.

4.3 Output

In each time step the plotted passengerformation is as a frame added to a moviestream. After the sumulation finished the result can be watched as ".avi"-moviefile.

5 Simulation Results and Discussion

For the sake of analyzing diffrent scenarios in simulations the system is built up (mostly) deterministic. The only random generated value is the weigth of each passenger. A (more) real system would embody much more uncertanties. For example the decay rate and the amplification of the social force would vary due to diffrent cultural backgrounds or behaviours. This is similarly true for the parameter tuple for the physical force where varying volumina of people would lead to varying parameters but not necessarily corresponding masses.

5.1 Parameter identification

The aim of the first simulations was to figure out phisically correct parameters. And there were alot of parameters to define among them obvious ones like the size of one time step dt or the wheigthing of each force $\vec{F_i}$. But on the other hand there are such subtle parameters as the decay rates of the wall-, social-, and pysical-forces or the radius of influence for these forces.
For a "smooth" simulation a small step size close to zero would be optimal. But for the sake of computational effort a value of $dt = 0.3$ is choosen. Other variables regarding smoothnes are decayrate of forces and radius of influence of these forces. If a passenger enters an area of influence in which it suddently will experience a strong change of forces, the passenger could behave "instable" in the sense of being pushed through walls.
If those two tuples are set the other parameters can be choosen relarively freely.

5.2 Influence of Social force

The social force model was introduced to describe the behaviour of pedestrians on pavement or in rooms. Airplanes could be considered very small rooms or better corridors. Beforehand there was the concern that this force applied on alot of passengers concentrated in a small area could lead to an unnatural behaviour during simulation. This turned out to be partially true.

As the social force tendencially unables a passenger to follow its shortest path to an exit one can expect a longer lasting evacuation than whitout an implemented force. Several runs of the simulation confirmed this.
The intoduction of the social force let also to a more "decent" behaviour of the passengers. The passengers lined up in a row and fewer tries to overturn others were attempted. This can be seen in Figure 9 and Figure 10.

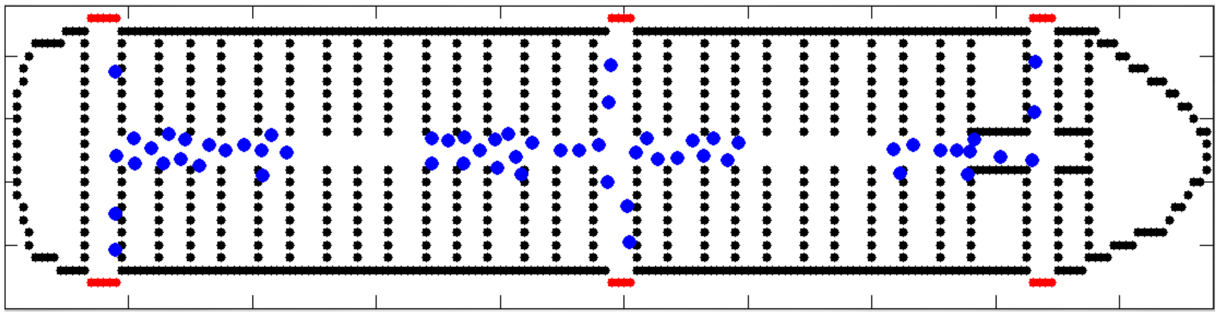


Figure 9: Simulation without social force

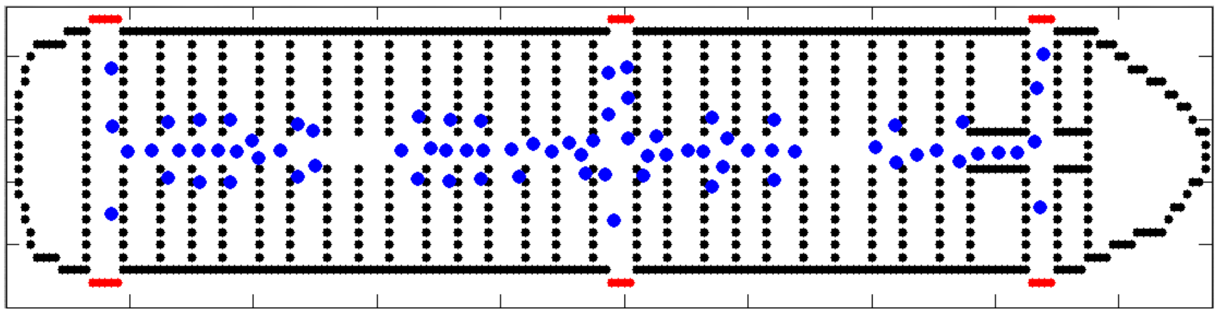


Figure 10: Simulation with social force

5.3 Simulated situations

There were different simulations performed to analyze and verify the model. Two different airplanes were simulated: A smaller Embraer 190 with 100 passengers and an Airbus 380. For the Airbus, only the lower economy class deck with 402 passengers was simulated.
Usually, bigger aircrafts are more difficult to evacuate simply because of the many passengers to be brought out of the airplane within the required 90 seconds. To see the differences, the smaller Embrear 190 and the biggest passenger jet of the world, the A380 were simulated.

Figure 11: Embraer 190 and Airbus A380, Source: (4)

5.3.1 Airbus A380

Figure 12 shows the model of the Airbus A380. Flight attendants marked with green were placed either at the exits or in the corridor. At the exits where no flight attendant is, the area is marked yellow which means that passengers there hesitate to jump and therefore walk slower within this area.

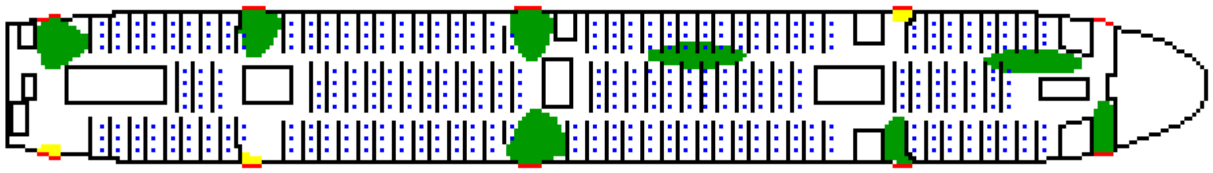


Figure 12: A380 Model

5.3.2 Embraer 190

For the model of the smaller Embraer 190, two simulations were performed. The initial situation is shown in Figure 13. In the first simulation (SIM 1), the flight attendants were placed in the corridor. For the second simulation (SIM 2), they were placed next to the emergency exit on the left side of the plane. At exits where no flight attendant is, the area is again marked yellow meaning that the passengers walk slower there.

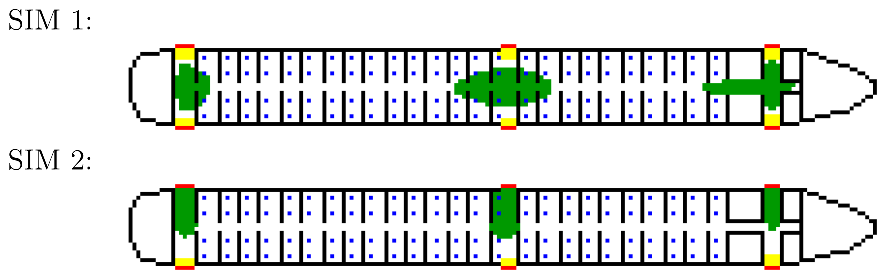


Figure 13: Embraer 190 Model

5.4 Results

5.4.1 Positioning of the flight attendants

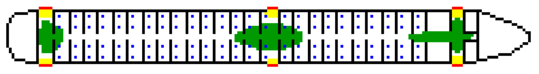
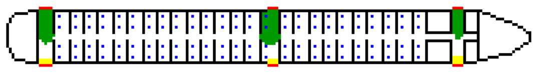
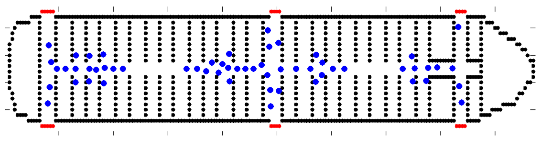
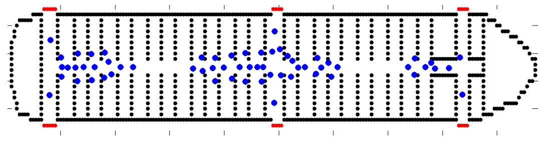
SIM 1	**SIM 2**
	
Evacuation time: 360 steps	Evacuation time: 428 steps
Frame No. 108: 	Frame No. 108: 

Figure 14: Embraer 190 results

Figure 14 compares the simulated situation for the Embraer 190. The simulation SIM 1 takes 360 steps compared to 428 steps for SIM 2. This means that the evacuation takes about 1.4 times longer in SIM 2. The reason for that is obvious to see in Frame 108. The flow in SIM 1 is heavily improved by the presence of a flight attendant in the corridor in the center of the plane. There are about 56 passengers willing to leave the plane through the exits in the middle. The exits themselves are much wider than the corridor of the plane. The most critical point is at the center of the plane where passengers turn to the exits.
The presence of flight attendants optimises the flow at this point as seen in SIM 1. The passengers in SIM 1 are grouped more in a column than in a crowd which means that they don't obstruct each other too much. This results in a constant flow and hence a faster evacuation.
The evacuation of the Airbus A380 is shown in Figure 15. It showed up that the

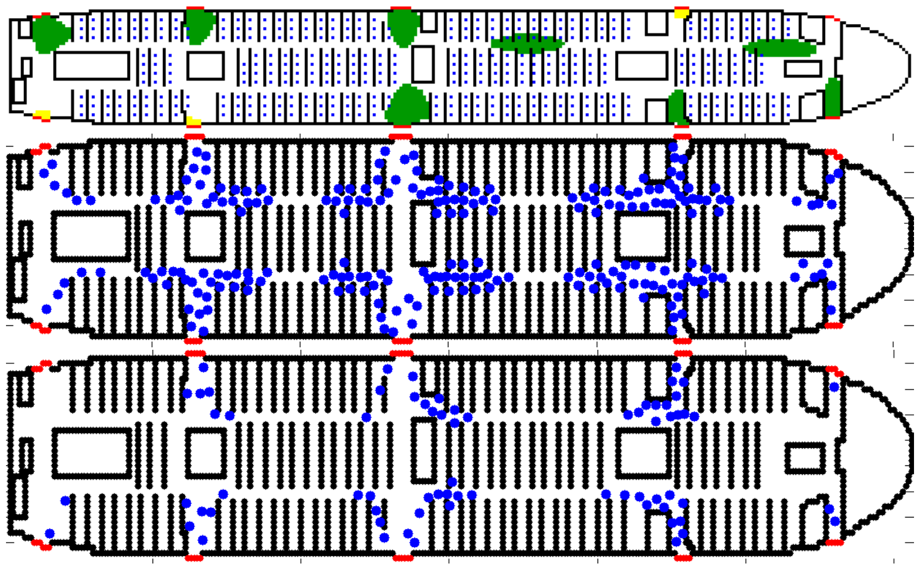


Figure 15: A380 Simulation frame No. 120 and frame No. 236

center exits had to swallow the most passengers. The last passenger that leaves the rear exit does that after 220 time steps while the last passenger over all leaves the plane after 428 time steps through the second exit from the front.
By looking at frame No. 120 and 236 shown in Figure 15, one can see that a flight attendant at a wide exit has hardly any effect, whereas a flight attendant at the second exit from the front on the right really brings a benefit. The passengers leaving the plane through this exit are faster than on the left side.

As already seen in the simulation of the Embraer 190, the flight attendants are the most efficient if they were placed at bottle necks.

5.4.2 Hesitation at the exits

The question was raised whether the social force model is reliable to simulate the situation in which passengers hesitate to jump down the slides at the exits. In nearly all simulated situations, the most critical parts were not the exits themselves, but the narrow corridors or corners. It showed up that the flow got slow always where passengers of many rows had to leave through a narrow bottle neck. Even with more hesitation time at exits meaning slower speed of passengers in yellow zones, the problem kept at the narrow bottle necks. The social force model is suitable to simulate this situation although it might be slightly different to the reality at the exits. The much bigger problem showed up at narrow bottlenecks. Eventually, the construction of the airplane seemed to be the much bigger problem then the aspect of hesitation with respect to the efficiency of evacuation.

5.5 Comparison to real live experiments

If one analyzes the real live experiments made with an Airbus A380 (5) that there are several diffrences to the resulted simulations.
Firstly, some of the personel in the experiment does not have a constant position as in the simulation. Also there is an attendant at each exit preventing a "bottleneck" situation. Further on, one can conclude that real live personal can intervene more diversly than simulated staff.
But apart from the dynamic behaviour of the staff their influence is very similar to the simulated one. Also the behaviour of the passengers diverges very little admittedly with fewer influence as expected of social force attributes.

6 Summary and Outlook

6.1 Identification of parameters

The parameters mentiond in 5.1 can be further optimized to complete the model. Further on, the implementation of individually behaviours of forces could lead to a more realistic model.

6.2 Adapted social force

Another possible extention could a speed dependent social force model be. Similarly to cars - where one also tries to have a free region of space in front of/around one to feel comfortable in - the effect could weaken at increased speeds e.g. when fluently driving on the autobahn fewer space is tolerated.

6.3 Simulation

The simulation showed that the social force model suits well for the simulation of an airliner. The effect of hesitation at the exits was overestimated. It showed up that the flow of people is mostly disturbed by bottle necks due to the construction of the aircraft. If flight attendants were placed there, the flow could be optimised which resulted in an overall better performance.
For further simulations, an interesting field of studies would be to perform tests with different shapes of corridors and eliminated bottle necks. For airplane manufacturers, there is always the balancing act between building jets carrying as many passengers as possible and safety aspects.

6.4 Conclusions

Over all one can say that the fundamental questions raised in **??** were answerded by running simulations described in **??** and in **??**.
Also one refered to the expected results stated in **??** by comparing real live experiments with the gained insigths of the simulation in 5.5. The analysis of the influence of the personal is stated in section **??**.

References

[1] D. Helbing, L. Buzna, A. Johansson, T. Werner. *Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solution*. informs, Germany, 2005

[2] Mohecine Chraibi. *Systeme gewöhnlicher Differentialgleichungen zur Beschreibung von Fussgängerdynamik*. Diplomarbeit, Technische Universität Hamburg, November 2008

[3] Picture Souce: `http://www.wikipedia.ch`

[4] Picture Souce: `http://www.time.com`

[5] `http://www.youtube.com/watch?v=_gqWeJGwV_U&feature=related`

A Research Plan

MATLAB FS11 – Research Plan

Document Version: 2
Group Name: lost pedestrians
Group participants names: Bühler Lukas, Heer Philipp

COMMENTS:

- Overall, the plan is interesting and well structured. It tries to answer more real-world, rather than fundamental questions, but it is perfectly fine.
- The only thing tyou could add is an explicit reference to the benchmarks you want to validate your model against (eg. the 90 seconds?)

General Introduction

In January 2009 an Airbus was forced to land in the Hudson River in New York. At this event 150 persons had to leave the plane before it sank to the ground of the river.
In this project the evacuation of airplanes is analyzed. After a crash it is possible for an airplane to explode due to ignited fuel or after a water landing to sink. During such an emergency situation it is important to evacuate all passengers and crew members of an airplane in a very short amount of time in order to maintain their survival.
Evacuation scenarios were studied before. Emerging fires or earthquakes lead to similar situations in buildings. Even the evacuation of airplanes has already been researched because hundreds of lives could be at stake. The "International Civil Aviation Organization" (ICAO) limited the maximum time of evacuation to 90 seconds for each plane. This time reference has to be proved by each manufacturer before a plane can go into service.

Fundamental Questions

The aim is to implement the social force model and to simulate the evacuation of an airplane. In specific, the following question should be answered:

1) Is this model suitable for the evacuation situation of an airplane or should there be done changes in order to get reliable results? Example: Do people hesitate to jump down the slides at the exits and does the social force model fail there?
2) Could suitable instructions of the cabin personal be simulated in the model and how does it affect the performance of the scenario?

Expected Results

1) There is data of evacuation experiments with real aircrafts available. This will help to verify and test the model. The answer of the question will be the confirmation of the test results by the model or a succession to improve the model. If there are differences of our model to the original social force model, they should be pointed out and explained.
2) We expect the instructions of the personal to be a positive effect on performance. To simulate this, there will be changes in the original social force model.

References

Paper provided by the course: Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solutions

Diplomarbeit TU Hamburg: Systeme gewöhnlicher Differentialgleichungen zur Beschreibung von Fußgängerdynamik, Mohcine Chraibi 2008

Research Methods

Continuous Modeling by implementing the social force model.

The model time step dt will be defined. The performance of the model will be measured by counting the time steps. By counting the amount of time steps, the model is validated relatively and can not be compared to a real time value. Nevertheless, this allows us to compare adaptations of our model.

B Matlab program code

B.1 pedest.m

```
%Lecture with Computer Exercises
%Modelling and Simulating Social Systems

%Projects: Pedestrian Dynamics
%Lukas Bühler Heer Philipp
%file: pedest.m

%run matrix description
 run init

 mov1 = avifile('planeS_1new.avi');

%time loop
for t=0:dt:T
    %reset seen phy_, obi_& soc_forces
    pasMat(1:NrOfpassenger,8:11)=0;
    pasMat(1:NrOfpassenger,17:20)=0;

    %passenger loop
    for i=1:NrOfpassenger
        if  pasMat(i,3)==1

        %calculation integer matrix indices
        xint=real(int16(pasMat(i,1)));
        yint=real(int16(pasMat(i,2)));
        %errorcatcher
        if xint<=0 || xint >= n || yint <=0 || yint >=m
            pasMat(i,3)=0;
        end
        end
        if  pasMat(i,3)==1
        %passenger in exit-area
        if  f(yint,xint)==Inf || f(yint+1,xint)==Inf ||f(yint-1,xint)==Inf || f(yint+2,xint)==Inf ||f(yint-2,xint)==Inf
            pasMat(i,3)=0;
            pasMat=sortrows(pasMat,3);
            passengerfleed=passengerfleed+1;
        end

        %passenger in "about to leave the plane"-area
        if f(yint,xint)==3 && crewflee==0
            pasMat(i,12)=FX(yint,xint)*fear_force_factor;
            pasMat(i,13)=FY(yint,xint)*fear_force_factor;

        %passenger in flightattendant-area
        elseif f(yint,xint)==6 && crewflee==0
            pasMat(i,12)=FX(yint,xint)*flee_force_factor;
            pasMat(i,13)=FY(yint,xint)*flee_force_factor;

        %passenger in no special area
        else
            pasMat(i,12)=FX(yint,xint)*force_factor;
            pasMat(i,13)=FY(yint,xint)*force_factor;
        end
    e_a=pasMat(i,12:13)/norm(pasMat(i,12:13)+[epsilon 0]);

    %influence of other passenger
    for j=1:NrOfpassenger
        if i~=j && pasMat(j,3)==1
            fv=[pasMat(i,1)-pasMat(j,1);pasMat(i,2)-pasMat(j,2)];
            if norm(fv)<pas_soc_influence_area
                n_ab=(pasMat(i,1:2)-pasMat(j,1:2))./norm(pasMat(j,1:2)-pasMat(i,1:2));

                %calculation f_phy
                if norm(fv)<pas_phy_influence_area
                    pasMat(i,8:9)=pasMat(i,8:9)+A_phy*exp((1-norm(pasMat(i,1:2)-pasMat(j,1:2)))/B_phy).*n_ab;
                end

                %calculation f_soc
                %phi_alphabeta = angle between passenger i and j
                phi_alphabeta=acos(dot(e_a,n_ab));
                pasMat(i,10:11)=pasMat(i,10:11)+((lamda+(1-lamda)*(1+cos(phi_alphabeta))/2))*A_soc*exp(1-norm↙
(pasMat(i,1:2)-pasMat(j,1:2))/B_soc).*n_ab;
            end
        end
    end

    %influence of wall_objects
    for j=1:NrOfObi
        %fv=vector between pas and wallelement
        fv=[pasMat(i,1)-obiMat(j,1);pasMat(i,2)-obiMat(j,2)];

        if norm(fv)<wall_influence_area
            wall_force=A_wall*exp(-norm(fv)/B_wall);

            if abs(obiMat(j,19))< abs(obiMat(j,20)) % wall in y-direction
                if obiMat(j,20)==[2.2] %only a wallelem above
                    if pasMat(i,2)< obiMat(j,2)
                        pasMat(i,12)=pasMat(i,12)+3*FX(yint-1,xint)*force_factor;
                    else
                        pasMat(i,17)=pasMat(i,17)+1.5*sign(fv(1))*wall_force;
                    end
                    pasMat(i,18)=-100;

                elseif obiMat(j,20)==[-3.3] %only a wallelem below
                    if pasMat(i,2) > obiMat(j,2)
                        pasMat(i,12)=pasMat(i,12)+3*FX(yint+1,xint)*force_factor;
                    else
                        pasMat(i,17)=pasMat(i,17)+1.5*sign(fv(1))*wall_force;
                    end
                    pasMat(i,18)=100;

                else
                    pasMat(i,17)=pasMat(i,17)+1.5*sign(fv(1))*wall_force;
                end

            else % wall in x-direction
```

```
                if obiMat(j,19)==[2.2] %only a wallelem on the rigth
                    if pasMat(i,1)< obiMat(j,1)
                        pasMat(i,13)=pasMat(i,13)+3*FY(yint,xint-1)*force_factor;
                    else
                        pasMat(i,18)=pasMat(i,18)+1.5*sign(fv(2))*wall_force;
                    end
                    pasMat(i,17)=-100;

                elseif obiMat(j,19)==[-3.3] %only a wallelem on the left
                    if pasMat(i,1) > obiMat(j,1)
                        pasMat(i,13)=pasMat(i,13)+3*FY(yint,xint+1)*force_factor;

                    else
                        pasMat(i,18)=pasMat(i,18)+1.5*sign(fv(2))*wall_force;
                    end
                        pasMat(i,17)=100;

                else
                    pasMat(i,18)=pasMat(i,18)+1.5*sign(fv(2))*wall_force;
                end
            end
            break;%only look at one wall-obj
        end
    end

    else
        pasMat(i,8:15)=0;
        pasMat(i,4:5)=0;
    end
end

%calculate f_tot
pasMat(1:NrOfpassenger,14:15)=(pasMat(1:NrOfpassenger,8:9)+pasMat(1:NrOfpassenger,10:11)+...
    pasMat(1:NrOfpassenger,12:13)+pasMat(1:NrOfpassenger,17:18)+pasMat(1:NrOfpassenger,19:20));

%calculate x'' and x'
pasMat(1:NrOfpassenger,4:5)=dt.*pasMat(1:NrOfpassenger,14:15)./[pasMat(1:NrOfpassenger,16) pasMat(1:↙
NrOfpassenger,16)];

%calculate x
pasMat(1:NrOfpassenger,6:7)=pasMat(1:NrOfpassenger,1:2);
pasMat(1:NrOfpassenger,7)=pasMat(1:NrOfpassenger,7)+epsilon;
pasMat(1:NrOfpassenger,1:2)=dt.*pasMat(1:NrOfpassenger,4:5)+pasMat(1:NrOfpassenger,1:2);
pasMat(1:NrOfpassenger,4)=pasMat(1:NrOfpassenger,4)+epsilon;
allMat=[pasMat;obiMat2];

%newplot
plot(allMat(NrOfpassenger+1:NrOfpassenger+NrOfObi2,1),allMat(NrOfpassenger+1:NrOfpassenger+NrOfObi2,↙
2),'.k','MarkerSize', 20);
hold on
plot(exiMat(:,1),exiMat(:,2),'.r','MarkerSize', 20);
plot(allMat(1+passengerfleed:NrOfpassenger,1),allMat(1+passengerfleed:NrOfpassenger,2),'.bl','MarkerSize', 30);
xlim([000 n]);
ylim([0 m]);
```

```
    %handle movie
    FF1 = getframe(gca);
    mov1=addframe(mov1,FF1);

    clf('reset')

    %if all passengers are gone -> allow crew to flee
    if sum(pasMat(1:NrOfpassenger,3))==0
        break;
    end
end
%end movie
mov1=close(mov1);
```

B.2 init.m

```
%Lecture with Computer Exercises
%Modelling and Simulating Social Systems

%Projects: Pedestrian Dynamics
%Lukas Bühler Heer Philipp
%file: init.m

%Init
clear all;
clc

%variable definition
pas_phy_influence_area=3;
pas_soc_influence_area=10;
att_soc_influence_area=10;
A_phy=80;
A_soc=40;
A_wall=1000;
B_phy=1;
B_soc=2;
B_wall=.5;
lamda=0.2;
dt=0.3;
epsilon=1e-10;
T=200/dt;
wall_influence_area=1.2;
wallrelevance_area=10;
heavy=1e10;
fear_force_factor=20;
force_factor=40;
flee_force_factor=60;
walelem=1;
paselem=1;
exielem=1;
crewflee=0;
passengerfleed=0;
gone=0;

%allocation of matrices
pasMat=zeros(10,20);
obiMat=zeros(700,20);
obiMat2=[];

%read picture/generate forcefield
 f = getFile();
[FX,FY]=computeGradientField1(f);
[m n]=size(FX);

%integration into matrices
for mm=1:m
    for nn=1:n
        %pas integration
        if f(mm,nn)==2;
            pasMat(paselem,2)=mm;
            pasMat(paselem,1)=nn;
            paselem=paselem+1;
        end

        %wall integration
        if f(mm,nn)==0;
            obiMat(walelem,2)=mm;
            obiMat(walelem,1)=nn;
            walelem=walelem+1;
        end
        %exit integration
        if f(mm,nn)==Inf;
            exiMat(exielem,2)=mm;
            exiMat(exielem,1)=nn;
            exielem=exielem+1;
        end
    end
end


%IT'S ALIVE MUAHAHAH!!
pasMat(:,3)=1;

%'previous' place
pasMat(:,6)=pasMat(:,1);
pasMat(:,7)=pasMat(:,2)+epsilon;
pasMat(:,4)=epsilon;
[NrOfpassenger entries]=size(pasMat);
[NrOfObi Obientries]=size(obiMat);
%random weigth
pasMat(:,16)=unidrnd(70,NrOfpassenger,1)+50;
obiMat(:,16)=heavy;

%set walldirection
for mn=2:NrOfObi

       %there is a wallelem above
   if f(obiMat(mn,2)+1,obiMat(mn,1))==0
       obiMat(mn,20)=[2.2];
   end
       %there is a wallelem below
   if f(obiMat(mn,2)-1,obiMat(mn,1))==0
       obiMat(mn,20)=obiMat(mn,20)-3.3;
   end
       %there is a wallelem on the rigth
   if f(obiMat(mn,2),obiMat(mn,1)+1)==0
       obiMat(mn,19)=[2.2];
   end
       %there is a wallelem on the left
   if f(obiMat(mn,2),obiMat(mn,1)-1)==0
       obiMat(mn,19)=obiMat(mn,19)-3.3;
   end
end
```

```
%simplify obiMat -> leave out useless wall elements
for j=1:exielem-1
    obiMat1=obiMat;
    [NrOfObi Obientries]=size(obiMat);
    gone=0;
    for i=1:NrOfObi
        fv=abs([obiMat1(i-gone,1)-exiMat(j,1);obiMat1(i-gone,2)-exiMat(j,2)]);
        NrOfObi=NrOfObi-gone;
        if abs(obiMat1(i-gone,19)) > abs(obiMat1(i-gone,20))
        if abs(norm(fv)) > wallrelevance_area
            obiMat1(i-gone,:)=[];
            gone=gone+1;
        end
        end
    end
    obiMat2=[obiMat2;obiMat1];
end

obiMat3=unique(obiMat2,'rows');
clear obiMat2;

[NrOfObi Obientries]=size(obiMat);
for i=1:NrOfObi
    if abs(obiMat(i,19)) > abs(obiMat(i,20))
        if obiMat(i,2) < 7 || obiMat(i,2) > m-7
        else
            obiMat3=[obiMat3;obiMat(i,:)];
        end
    end
end
obiMat2=obiMat;
obiMat=obiMat3;
clear obiMat4;
clear obiMat1;
clear obiMat3;
[NrOfObi Obientries]=size(obiMat);
[NrOfObi2 Obientries2]=size(obiMat2);
```

B.3 getFile.m

```
24.05.11 08:48  /Users/lukasbuehler/ETH/Sem2/Simulat.../getFile.m   1 of 1

function [F] = getFile()
%getFile: Convertion of an bmp-image to a matrix
%   This function converts a bmp-file to a matrix and returns it. There↙
are
%   only the following colors with their corresponding interpretation
%   allowed:
%
%   Color          Hex triplet    Description
%   White          FFFFFF         Space in which the passengers can freely
%                                 walk
%   Black          000000         Walls
%   Red            FF0000         Emergency exit
%   Blue           0000FF         Every blue pixel is recognised as a
%                                 pessanger
%   Light green    00FF00         Every light green pixel is recognised as↙
a
%                                 flight attendant
%   Dark green     009900         Zone which is influenced by the flight
%                                 attendant
%   Yellow         FFFF00         Special zones, in which pessangers
%                                 struggle to continue walking
%
%   The output matrix contains the following entries:
%   0=wall, 1=space, 2=passenger, 3=hesitation area, 4=flightattendant,
%   6=flightattendantarea, Inf=emergency exit


exit=0;
while exit==0
    [FileName,PathName] = uigetfile('*.bmp', 'Select a Bitmap File')
    I=imread(strcat(PathName,FileName));
    exit=1;
    if (find(I>6))
        exit=0;
        uiwait(msgbox('Wrong file'));
    end
end

space=find(I==5);
goSlow=find(I==4);
exit=find(I==2);
passenger=find(I==1);
flightattendant=find(I==3);
flightattendantarea=find(I==6);
wall=find(I==0);
[n,m]=size(I);
F=zeros(n,m);
F(space)=1;
F(goSlow)=3
F(exit)=Inf;
F(passenger)=2;
F(flightattendant)=4;
F(wall)=0;
F(flightattendantarea)=6;
F=flipud(F)

end
```

B.4 gradient_special.m

```
function [FFX FFY] = gradient_special(M)
%GRADIENT_SPECIAL Special gradient function which excludes matrix entries
%which are Inf
%   Input is a matrix m with a potential field and Inf entries↙
representing
%   walls. This function generates a vector field representing the↙
gradient
%   field of m, but ignores all Inf entries.
[a b]=size(M);

FX=zeros(a,b);
FY=zeros(a,b);
caseX=0;
caseY=0;

%Cases: W=Wall  °=Point to be handled
%  1.   W ° W
%  2.   W ° °


for m = 1:a    %y-direction
    for n = 1:b   %x-direction

        %X-Direction
        if (M(m,n)~=Inf) %Point is no wall element
            if(n>1 && n<b) %No element at the boarder of the matrix
                if(M(m,n-1)==Inf && M(m,n+1)==Inf)
                    caseX=1;
                elseif (M(m,n-1)==Inf)
                    caseX=2;
                elseif (M(m,n+1)==Inf)
                    caseX=3;
                else
                    caseX=4;
                end
            elseif (n<b) %Element at the lower boarder of the matrix
                if(M(m,n+1)==Inf)
                    caseX=1;
                else
                    caseX=2;
                end
            else %Element at the upper boarder of the matrix
                if(M(m,n-1)==Inf)
                    caseX=1;
                else
                    caseX=3;
                end
            end


        else %Point is a wall element
            if(n>1 && n<b) %No element at the boarder of the matrix
                if(M(m,n-1)==Inf && M(m,n+1)==Inf)
                    caseX=5;
                elseif (M(m,n-1)==Inf)
                    caseX=6;
                elseif (M(m,n+1)==Inf)
                    caseX=7;
                else
                    caseX=8;
                end
            elseif (n<b) %Element at the lower boarder of the matrix
                if(M(m,n+1)==Inf)
                    caseX=5;
                else
                    caseX=6;
                end
            else %Element at the upper boarder of the matrix
                if(M(m,n-1)==Inf)
                    caseX=5;
                else
                    caseX=7;
                end
            end

        end

        switch caseX
            case 1
                FX(m,n)=0;
            case 2
                FX(m,n)=(M(m,n)-M(m,n+1));
            case 3
                FX(m,n)=(M(m,n-1)-M(m,n));
            case 4
                FX(m,n)=(M(m,n-1)-M(m,n+1))/2;
            case 5
                FX(m,n)=0;
            case 6
                FX(m,n)=0;
            case 7
                FX(m,n)=0;
            case 8
                FX(m,n)=0;
        end

        %FX(m,n)=caseX;




        %Y-Direction
        if (M(m,n)~=Inf) %Point is no wall element
            if(m>1 && m<a) %No element at the boarder of the matrix
                if(M(m-1,n)==Inf && M(m+1,n)==Inf)
                    caseY=1;
                elseif (M(m-1,n)==Inf)
                    caseY=2;
                elseif (M(m+1,n)==Inf)
                    caseY=3;
                else
                    caseY=4;
                end
            elseif (m<a) %Element at the lower boarder of the matrix
                if(M(m+1,n)==Inf)
                    caseY=1;
                else
                    caseY=2;
                end
            else %Element at the upper boarder of the matrix
                if(M(m-1,n)==Inf)
                    caseY=1;
                else
                    caseY=3;
                end
            end


        else %Point is a wall element
            if(m>1 && m<a) %No element at the boarder of the matrix
                if(M(m-1,n)==Inf && M(m+1,n)==Inf)
                    caseY=5;
                elseif (M(m-1,n)==Inf)
                    caseY=6;
                elseif (M(m+1,n)==Inf)
                    caseY=7;
                else
                    caseY=8;
                end
            elseif (m<a) %Element at the lower boarder of the matrix
                if(M(m+1,n)==Inf)
                    caseY=5;
                else
                    caseY=6;
                end
            else %Element at the upper boarder of the matrix
                if(M(m-1,n)==Inf)
                    caseY=5;
                else
                    caseY=7;
                end
            end

        end

        switch caseY
            case 1 %  W ° W
                FY(m,n)=0;
            case 2 %  W ° °
                FY(m,n)=(M(m,n)-M(m+1,n));
            case 3 %  ° ° W
                FY(m,n)=(M(m-1,n)-M(m,n));
            case 4 %  ° ° °
                FY(m,n)=(M(m-1,n)-M(m+1,n))/2;
            case 5 %  I I I
                FY(m,n)=0;
            case 6 %  I I °
                FY(m,n)=0;
            case 7 %  ° I I
                FY(m,n)=0;
            case 8 %  ° I °
                FY(m,n)=0;
        end
        % Current Point: °    Wall: W   Infinity:I

        %FX(m,n)=caseX;
```

```
    end
end


%Normalization of the vector length
[a,b]=size(FX);

for m = 1:a
    for n = 1:b
        if (FX(m,n)~=0 && FY(m,n)~=0)
            FFX(m,n)=(FX(m,n)/(sqrt(FX(m,n)^2+FY(m,n)^2)));
            FFY(m,n)=(FY(m,n)/(sqrt(FX(m,n)^2+FY(m,n)^2)));
        elseif(FX(m,n)~=0)
            FFX(m,n)=(FX(m,n)/abs(FX(m,n)));
            FFY(m,n)=0;
        elseif(FY(m,n)~=0)
            FFX(m,n)=0;
            FFY(m,n)=(FY(m,n)/abs(FY(m,n)));
        end
    end
end


end
```

B.5 computeGradientField1.m

24.05.11 09:32 /Users/lukasbuehler/ET.../computeGradientField1.m 1 of 1

```
function [FX, FY] = computeGradientField1(F)
%UNTITLED2 This function computes the shortest path from every point to↙
the
%next emergency exit.
%   The input matrix F contains a space described in the file getFile.m.
%   This function calculates a vector field containing the shortest path
%   from every point in the space to the next emergency exit.

%Find the Exits
[rowE,colE,v] = find(F==Inf);

%Create a new Space with only 1(Space) and 0(Wall) as entries
[sx,sy]=size(F);
NewF=ones(sx,sy);
Wall=find(F==0);
NewF(Wall)=0;



Exits(1,:)=rowE';
Exits(2,:)=colE';

options.nb_iter_max = Inf;
[D,S] = perform_fast_marching(NewF, Exits, options);
[FX,FY] = gradient_special(D);

D(D==Inf)=0; % Make infinity entries of D to 0


%Plot Contour
contour(D);
hold on;

%Plot Vectorfield
quiver(FX,FY,0.1);
hold on;

%Plot Walls
[rowW,colW,v] = find(NewF==0);
for n=1:1:size(rowW)
   h = plot(colW(n),rowW(n), '.b'); set(h, 'MarkerSize', 10);
end


%Plot Exits
for n=1:1:size(rowE)
   h = plot(colE(n),rowE(n), '.r'); set(h, 'MarkerSize', 20);
end

end

end
```