ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Project Report

# Trails of Irchelpark
# How students walk.

Marius Buehlmann & Anton Beitler

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.

Anton Beitler

Marius Bühlmann

# Contents

# 1 Individual contributions

Our work is based on the papers of Dirk Helbling (HKM97), (HSKM97) and the report of (PN10), who did the same lecture one semester before. We developed an own code for the given mathematical formulation. Most of the ideas in the code were developed in teamwork during the lectures of "Lecture with Computer Exercises: Modelling and Simulating Social Systems with MATLAB". There were some model ideas in (PN10), which we implemented and improved in our code. Everything is declared in the chapter Implementation. For writing the report, we divided the work in two parts. Anton has done the model description and the code implementation, Marius has described the results , the summary and the outlook. The summary and the outlook, was discussed in the team before.

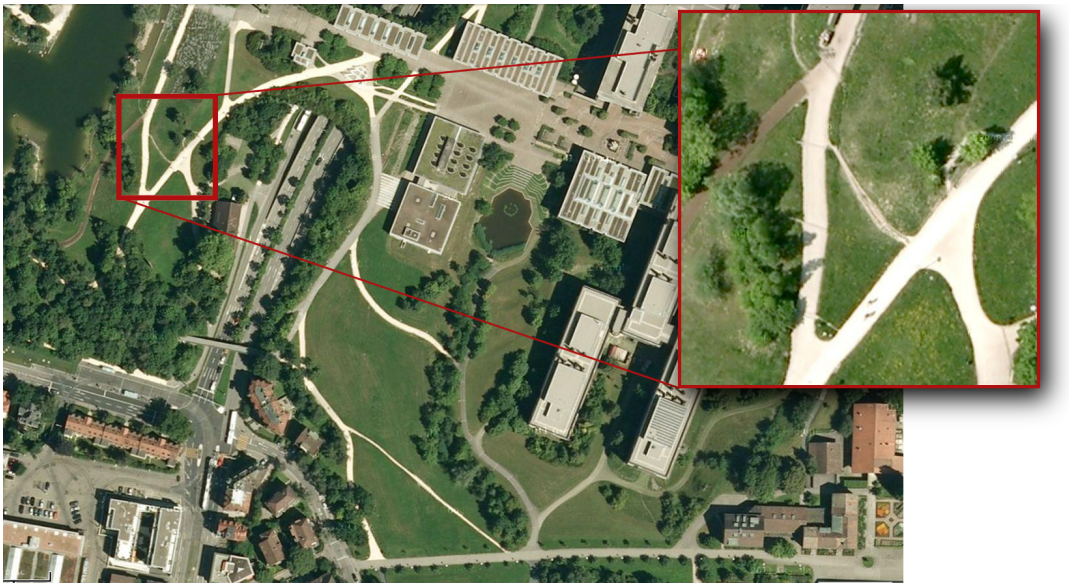


Figure 1: Example for a trail formation at Irchelpark, Zurich (Source: Google Maps)

## 2 Introduction and Motivations

We decided to contribute to the efforts of modeling the evolution of human trail systems. Such trails evolve apart from existing paths and often show characteristic patterns. The knowledge about which conditions lead to this phenomenon might be helpful in planning path-systems such as public green spaces efficiently. On the contrary there are situations where such trails need to be avoided for instance due to environmental protection where this concept might come handy, too.

Mainly we want to focus on the reproduction of known concepts and their results. Later we want to expand the knowledge a real-life problems on university campus of different complexity. Our goal is to verify or modify the model as required.

# 3 Description of the Model

## 3.1 Model requirements

Pedestrians can be viewed as moving entities that follow certain rules in choosing the direction of movement. From our own experience we know that the location of the destination may ultimately be factor that has the greatest impact in this decisionmaking. Nevertheless certain attributes of the current environment have an influence. Such attributes include for instance the existence and condition of walkways or the presence of obstacles.

These factors lead to a tradeoff situation between a shortest and a most comfortable path that has to be evaluated at each moment.

Additionally the model should consider environmental changes over time. For example some alteration of the ground's condition may be caused by weathering (heavy rain might cause muddy ground, aridity may dry out plants, etc.). But also each step being taken causes erosion and therefore changes the attractiveness of a certain path.

Thus a choice of a walking direction being made at a certain point of time depends on the choices of Pedestrians that walked this way before. Obviously this fact is essential for the evolution of trails and therefore a key requirement for our model.

## 3.2 Active Walker Model

The active walker model[1] is the model of choice when it comes to modelling the movement of pedestrians. The model builds laws of mutual interaction between one entity and it's environment. Though the interaction is of microscopic scale it indirectly influences other entities in the same environment. This fits perfectly to the requirements stated above as it comprises the bidirectional dependency of a pedestrian's walking direction and environmental change.

In order to embrace the state and evolution of a system consisting of pedestrians and the ground they are walking on, certain descriptive objects and parameters are introduced.

$\vec{r}_\alpha$**:** Position of Pedestrian $\alpha$

[1] (HSKM97, 2)

**Ground** $G(\vec{r},t)$**:** The comfort of walking on the ground at position $\vec{r}$ denoted by a scalar with initial value $G(\vec{r},0) = G_0(\vec{r})$ limited by a maximum comfort $G_{\max}(\vec{r})$.

**Durability** $T(\vec{r})$**:** The resistance of the ground towards erosion of any kind.

**Intensity** $I_\alpha$**:** Intensity of the footprints of a pedestrian.[2]

**Visibility** $\sigma(\vec{r}_\alpha)$**:** A scalar that indicates the distance up to which the condition of the ground is taken into account in choosing the next step.

**Potential** $V(\vec{r}_\alpha,t)$**:** A scalar characterizing attractiveness of walking at place $\vec{r}_alpha$.

### 3.3 Mathematical formulation

Having defined the characteristics of the systems to be modelled we now derive how they are suggested to evolve in time.

→ Environmental change

The change of the ground's comfort of walking is reflected in two dynamics: a *restoration* (to initial state) and an *erosion* (erosion of plants leading to a higher comfort) of the ground. The restoration depends on the durability $T(\vec{r})$ whereas erosion takes place due to the intensity of people's footprints. This leads to an ordinary differential equation of first order for the evolution of the ground's condition in time:

$$\frac{dG(\vec{r},t)}{dt} = \underbrace{\frac{[G_0(\vec{r}) - G(\vec{r},t)]}{T(\vec{r})}}_{\text{restoration}} + \underbrace{I(\vec{r})\left[1 - \frac{G(\vec{r},t)}{G_{\max}(\vec{r})}\right]\sum_\alpha \delta(\vec{r} - \vec{r}_\alpha(t))}_{\text{erosion}} \tag{1}$$

→ Walking potential The potential of a certain place $V(\vec{r}_\alpha,t)$ is defined as an integral over the ground function $G(\vec{r},t)$ weighted with a factor of exponential decay radial from the reference point $\vec{r}_\alpha$. The damping factor of the exponential is the visibility parameter $\sigma(\vec{r}_\alpha)$:

$$V(\vec{r},t) = \int G(\vec{r},t)\cdot \exp\left(-\frac{|\vec{r} - \vec{r}_\alpha|}{\sigma(\vec{r}_\alpha)}\right) d^2r \tag{2}$$

→ Determination of walking direction

[2] Though the proposed model states the Intensity as a function of space (see (HKM97)) we consider it more applicable to introduce it a constant.

The walking direction is influenced by the direction of the destination $\vec{d_\alpha}$ which is $\vec{e}_\alpha(\vec{r}_\alpha) = \frac{(\vec{d_\alpha} - \vec{r}_\alpha)}{|\vec{d_\alpha} - \vec{r}_\alpha|}$ and the gradient of the potential $\nabla_r V(\vec{r}_\alpha, t)$. The walking direction is the superposition of both:

$$\vec{e}_\alpha = \frac{\vec{d_\alpha} - \vec{r}_\alpha + \nabla_r V(\vec{r}_\alpha, t)}{|\vec{d_\alpha} - \vec{r}_\alpha + \nabla_r V(\vec{r}_\alpha, t)|} \tag{3}$$

The resulting trajectory $\vec{r}_\alpha(t)$ of a pedestrian $\alpha$ walking with speed $v_\alpha$ is therefore determined by

$$\frac{d\vec{r}_\alpha}{dt} = v_\alpha \vec{e}_\alpha(\vec{r}_\alpha, t) \tag{4}$$

# 4 Implementation

This section is devoted to describe how the model is put into action. In doing so we decided to simplify the model in some minor aspects:

- The durability $T(\vec{r})$ is a constant $T$.
- The intensity $I_\alpha = I$ is the same for every pedestrian.
- The visibility $\sigma(\vec{r}_\alpha)$ is a constant $\sigma$

The model is implemented in MATLAB which is optimized for operations matrices. As we aim to optimize our code for speed we choose a functional programming paradigm and try to avoid loops and conditional structures wherever possible.

In order to run a numerical simulation of the proposed model the equations have to be discretised as follows.

→ Environmental change

The values for ground's comfort $G(\vec{r}, t)$ are stored in a $m \times n$-matrix $G(k)$ such that $G(\vec{r}_{i,j}, uT) = G_{i,j}(k)$, where $\vec{r} = \begin{pmatrix} i \\ j \end{pmatrix}$ $(i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n;\ u \in \mathbb{N})$ becomes a discrete coordinate vector for a position of an element in the matrix. Throughout our work we chose $n = m$ and $T = 1$ such that $k = u$.

The ODE (1) then becomes an iteration law:

$$G_{i,j}(0) = G_0 \tag{5}$$

$$G_{i,j}(k+1) = G(k)_{i,j} + \frac{(G_{0,(i,j)} - G_{i,j}(k))}{T} \tag{6}$$

$$+ I \cdot \left(1 - \frac{G_{i,j}(k)}{G_{\max}}\right) \quad \forall i, j \in [m] \times [n] \tag{7}$$

The initial ground state can include a set of elements of value $G_{\max}$ which then indicates a pathway.

→ Walking potential

The potential is also a matrix of the same size as the ground. An element in this matrix is the evaluation of the Integral at a given position $\vec{r}_{i,j}$ where the integral over the given two dimensional space of the ground becomes a summation over all

Figure 2: Discretisation of a pathway

matrix elements:

$$V_{i,j}(k) = \sum_{o,p} G_{o.p}(k) \cdot \exp\left(-\frac{|\vec{r}_{o,p} - \vec{r}_{i,j}|}{\sigma}\right) \tag{8}$$

For better performance the evaluate this summation only every other iteration step without significant loss of accuracy.

→ Determination of walking direction

For choosing the resulting walking direction we used the same method as (PN10): We determine the direction of both, the destination $\vec{d}$ and the maximum potential $\vec{v} = \underset{i,j}{\operatorname{argmax}}\,(V_{i,j}(k))$ in the neighbourhood. The resulting direction is a superposition of both with a weight we call the importance $IMP$.

$$\vec{e} = IMP \cdot \frac{\vec{d} - \vec{r}_{i,j}}{|\vec{d} - \vec{r}_{i,j}|} + (1 - IMP) \cdot \frac{\vec{v} - \vec{r}_{i,j}}{|\vec{v} - \vec{r}_{i,j}|} \tag{9}$$

Since each matrix-element represents a position, one pedestrian has eight possible positions in the next iteration step. The position south $\vec{s} = (1, 0)$ servers as a reference direction for the calculation of an angle $\alpha$:

$$\alpha = \operatorname{sign}(e_2) \cdot \operatorname{acos}\left(\frac{\langle \vec{e}, \vec{s} \rangle}{|\vec{e}| \cdot |\vec{s}|}\right) + 2\pi \cdot \mathbb{I}\{-e_2 > 0\} \tag{10}$$

($\mathbb{I}\{M\}$ denotes the indicator function applied on a set $M$)

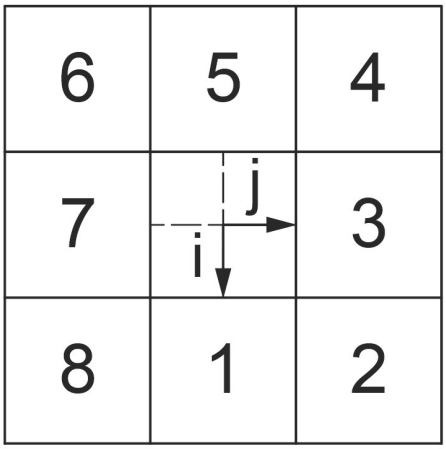



Figure 3: The eight sectors

Based on this angle $\alpha$ it is possible to determine which of the sectors $S_i$ in the direct neighbourhood of the current position is going to be taken.

$$s = \left( \lfloor \frac{\alpha + \pi/8}{\pi/4} \rfloor \mod 8 \right) + 1 \tag{11}$$

With this implementation we were able to replace many loops and conditional structures by algebraic operation which boost the performance significantly compared to (PN10)

# 5 Simulation Results and Discussion

## 5.1 Reproduction

In this part it is shown how the results of (HKM97) can be reproduced. The results are also compared with (PN10). In all our simulations a 50 by 50 square grid is used.

### 5.1.1 Triangle destinations

In this geometry the entry points and the destinations are located in a triangular geometry. The goal was to reproduce the results in (HKM97, Fig. 2). When we set the parameters as we calibrated them for the Irchelpark geometry, the pathway system looks as in the figure of (HKM97). But to reproduce a trail which looks like the 3rd of these figures, we had to change the importance and the intensity parameters. When we set the importance to 0.2 and the intensity to 0.3 the results were similar. Unfortunately we do not know the values of the parameters used in (HKM97). Rising the visibility parameter leads to a similar change in the pathway system as in the paper describe above. Between the left and the right bottom entry there are some other results. This might because of the grid to determine the walking direction as we implemented it and the importance might play a major role. If we set the importance to zero, which leads to a straight way to the center of the area, the pedestrians will stay and walk around in this center. They are not able to reach their destination.

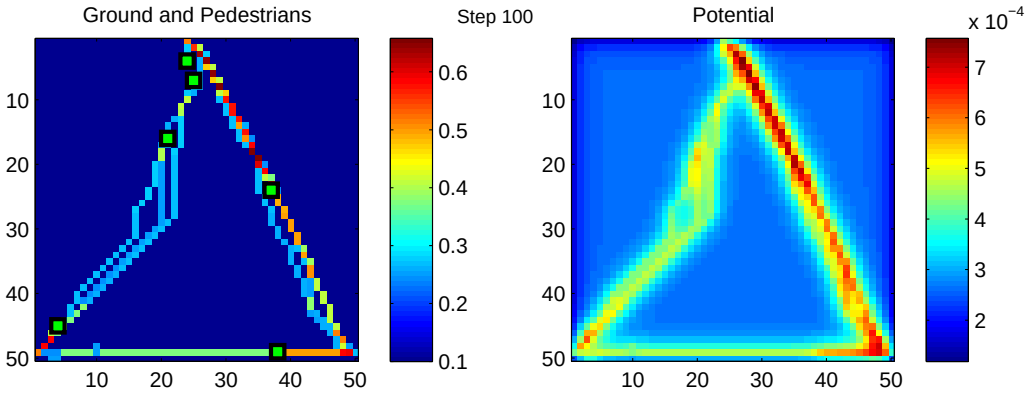



Figure 4: Vis=1.0, Imp=0.5, Int=0.2

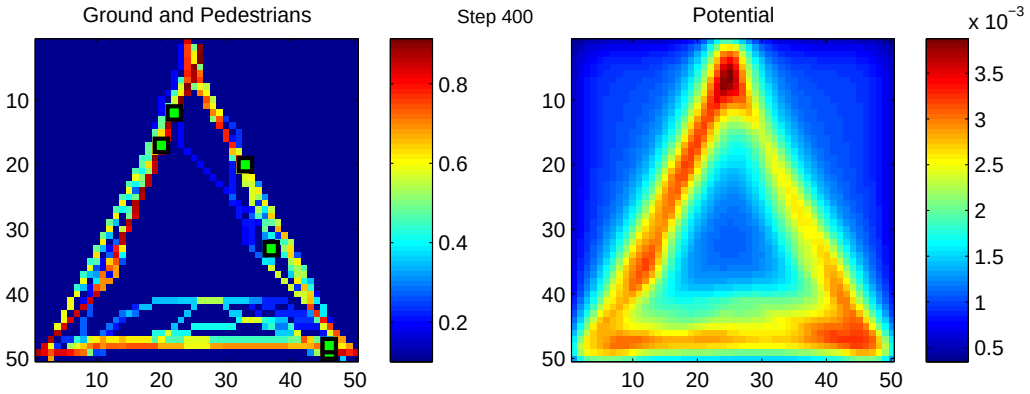



Figure 5: Vis=2.0, Imp=0.2, Int=0.3

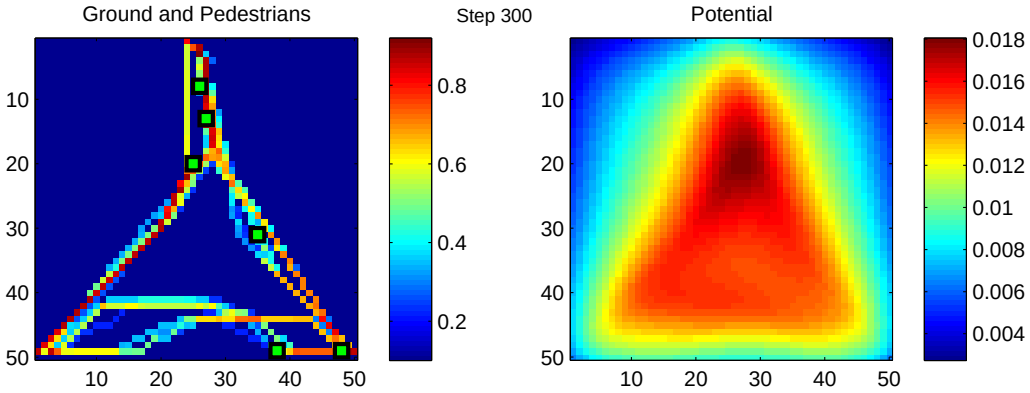



Figure 6: Vis=6.0, Imp=0.2, Int=0.3

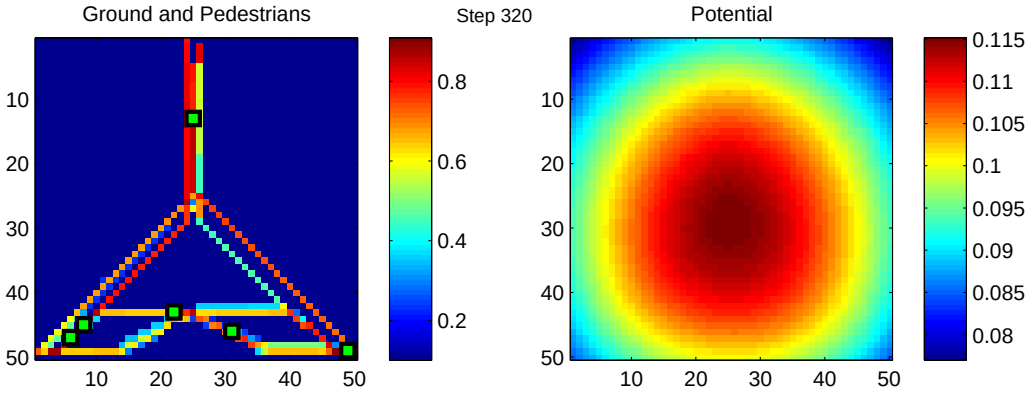



Figure 7: Vis=50.0, Imp=0.2, Int=0.3

### 5.1.2 Corner entry

In this part we tried to get the same result for the corner geometry as in (HKM97). To be plausible, we took the parameters we determined in the triangle geometry and did not change them. As shown in the figures below, in a manner it was possible to generate the same results. But like in the triangle geometry, the behaviour for a large visibility is not as smooth as it is expected. The guessed reason is described in the Summary chapter.

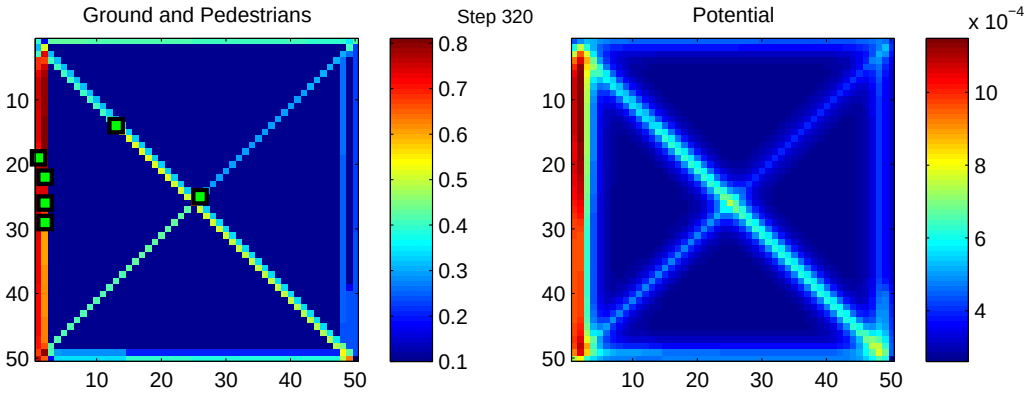



Figure 8: Vis=1.0, Imp=0.5, Int=0.2

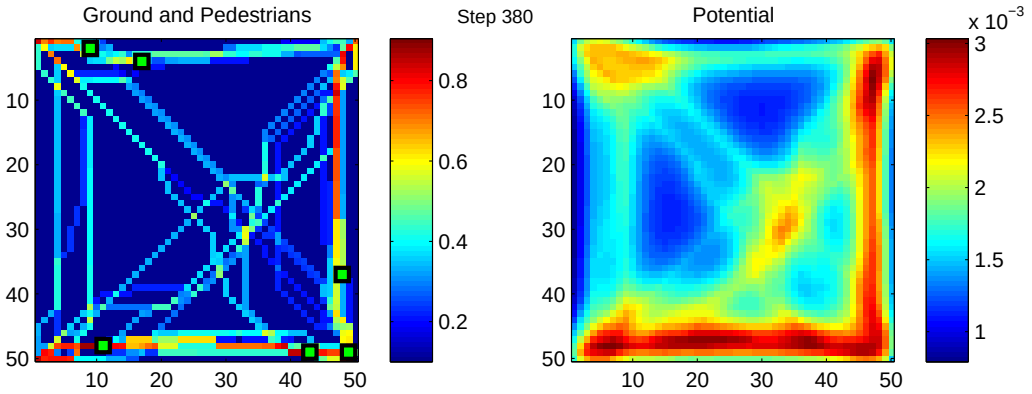



Figure 9: Vis=2.0, Imp=0.2, Int=0.3

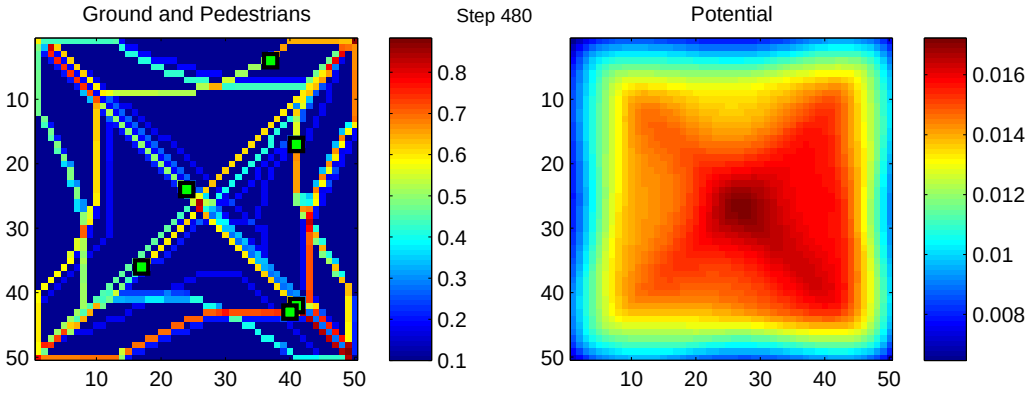



Figure 10: Vis=6.0, Imp=0.2, Int=0.3

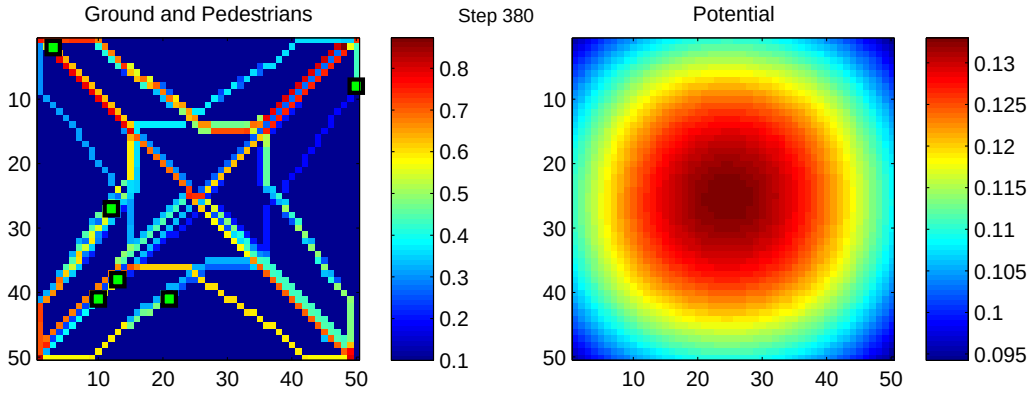
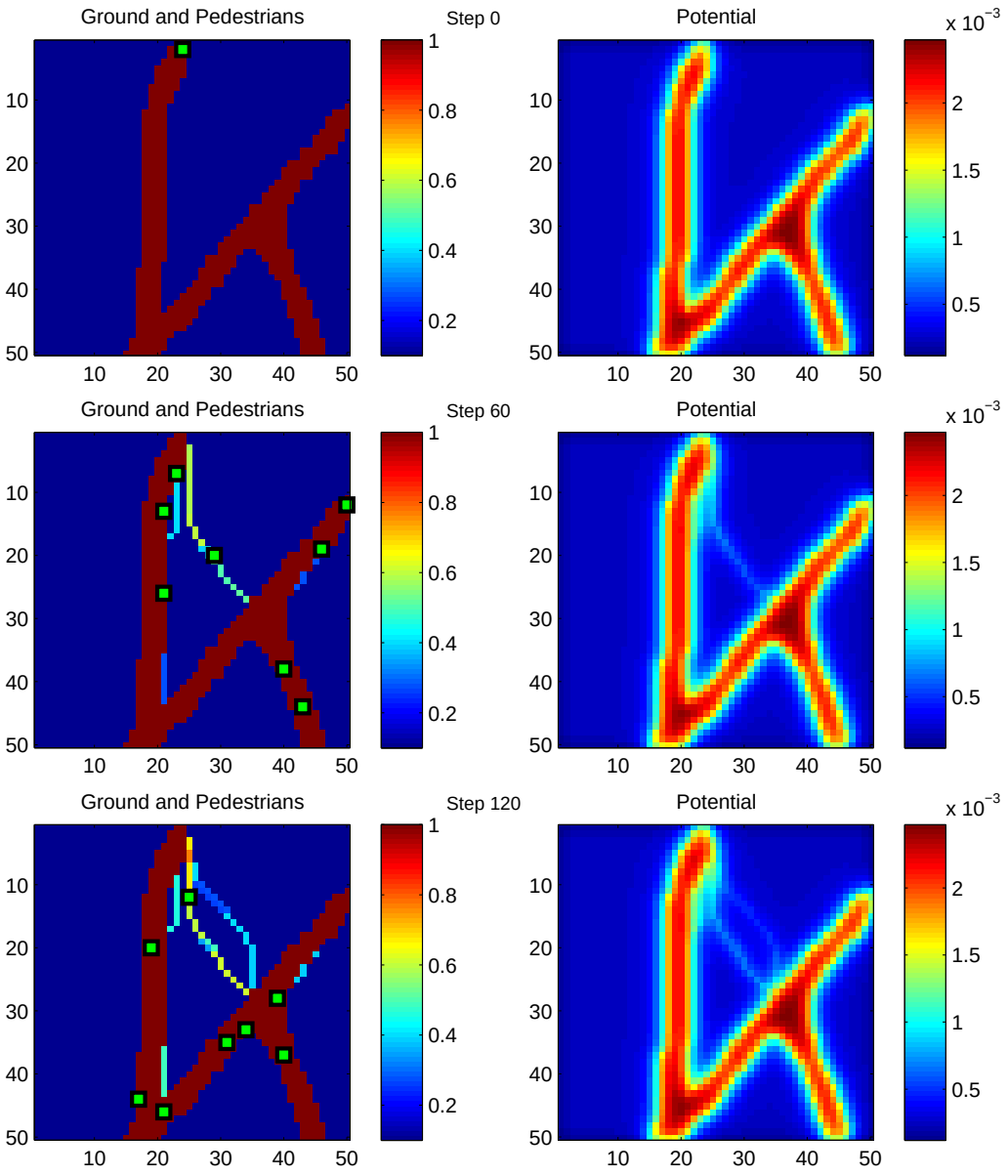
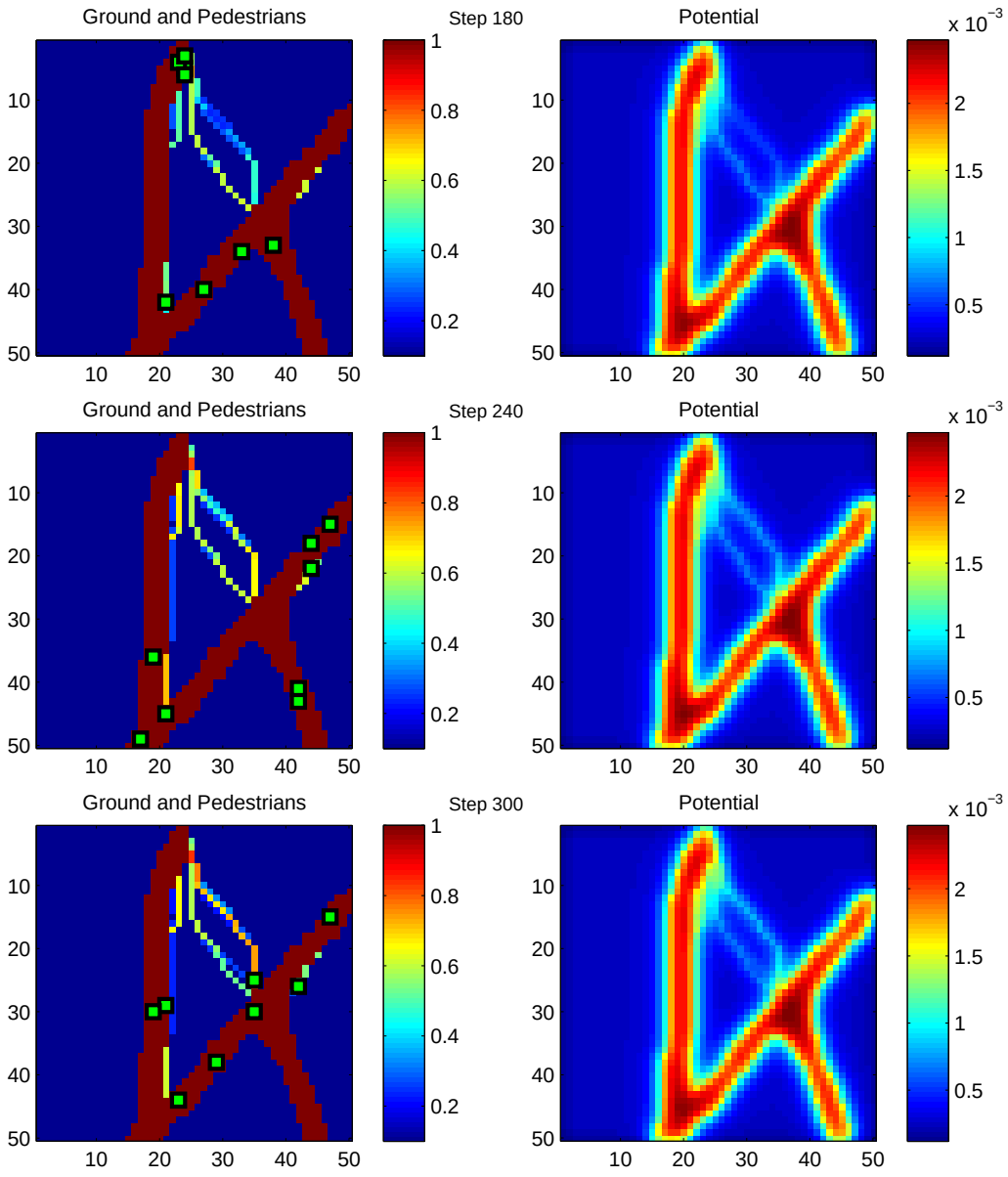


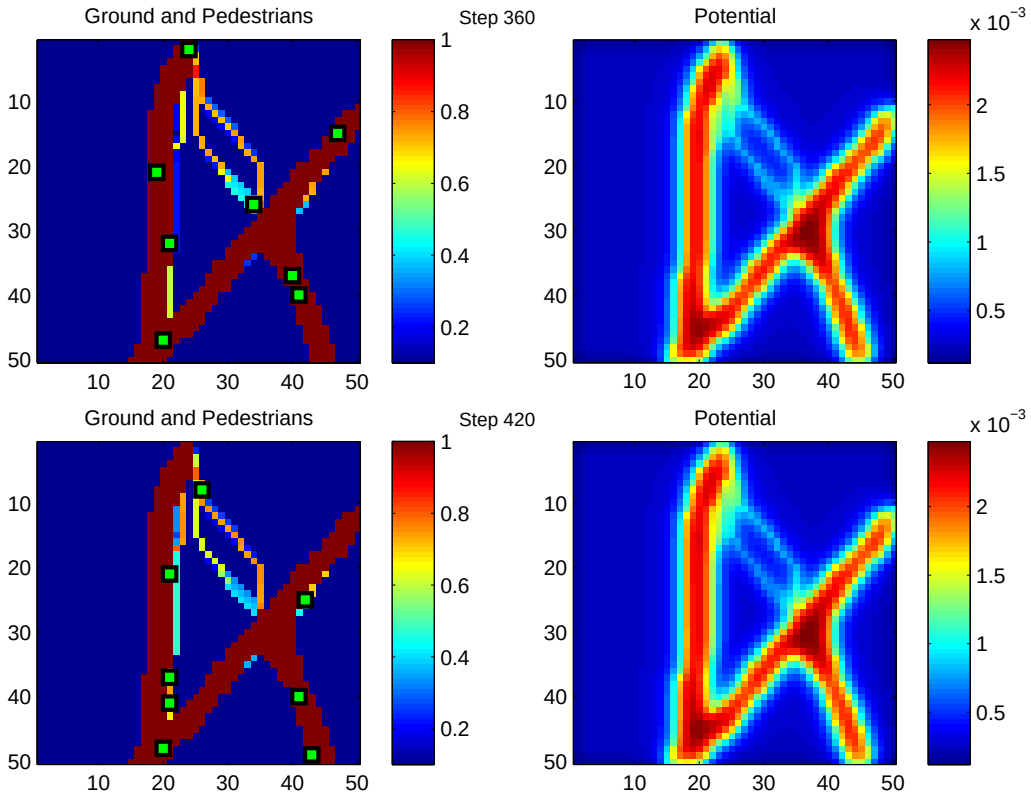

Figure 11: Vis=50.0, Imp=0.2, Int=0.3

## 5.2 Trails of Irchelpark

Our main goal in the research plan was to reproduce the pathway described in chapter 2. It was generated by students who did not want to do a loop way and walked directly over the grass.


Ground and Pedestrians
Step 0
Potential
x 10^-3
Ground and Pedestrians
Step 60
Potential
x 10^-3
Ground and Pedestrians
Step 120
Potential
x 10^-3


Ground and Pedestrians
Step 180
Potential
x 10^-3
Ground and Pedestrians
Step 240
Potential
x 10^-3
Ground and Pedestrians
Step 300
Potential
x 10^-3


Ground and Pedestrians
Step 360
Potential
x 10^-3
Ground and Pedestrians
Step 420
Potential
x 10^-3

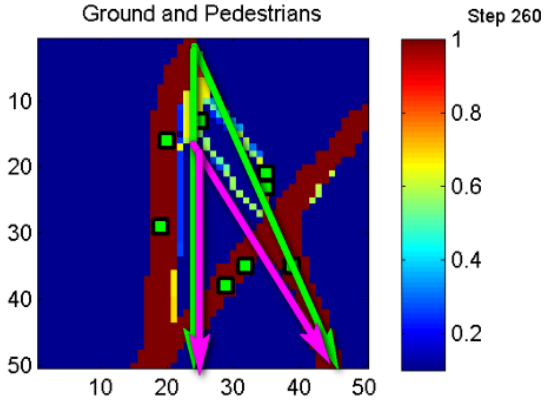



Figure 12: Direction

## 6 Summary

First of all, we were able to reproduce this trail nearby correct. All the pedestrians behave like expected an walk over the grass. The path is clearly recognisable on the plots. We also were able to reproduce the results from the paper (HKM97). In some cases there were some differences. But if you compare our results to the those o(PN10), it is interesting, that their model in some cases behaves as ours. Even they implemented some features in a different way. On the other hand, we used the idea of the importance and the implementation of the grid for the walking direction from (PN10). It might be, that those implementations cause these troubles. The angle between two directions is obviously to large. We can easily see this feature in the Irchelpark geometry, when pedestrians are walking from north to southeast. At the begin, the angel is small, this means the will walk straight down. While they walk south (green arrows), the angle between their walking direction and the destination is getting bigger. If 25 degrees are reached, the movement direction changes rapidly (purple arrows).

# 7 Outlook

We have realised that the implemented determination of the walking direction has its weakness. There are 8 possible walking directions in our model. If a pedestrian is walking in direction of south east, there are only 2 possible directions. We considered about a finer mesh and a pedestrian would move 5 squares every step. In such a model 32 different directions could be chosen, which would lead to a much more appropriate pathway systems. In addition, we were thinking about an importance which is depending on the angle between the direction on the vector of the biggest potential. If the biggest potential (usually this is on the pathway) is nearby in the same direction as the destination, the importance should be low. If the biggest potential is in a 90 degrees direction of the destination, the importance should have a maximum. In such a way, the pedestrians should behave more natural and walk on the path, if its direction is the same as the destination. Furthermore, to develop a more sensitive behaviour when 2 pedestrians cross would be a next challenge. People should realise at least 4 squares before they meet that somebody is coming and go aside. But to get better results, the first two suggestions for improvement seem to be more appropriate.

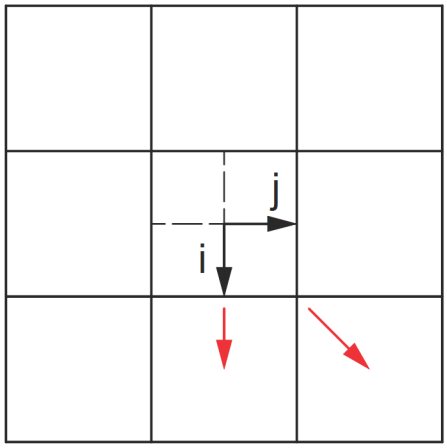



Figure 13: 2 possible directions south-east

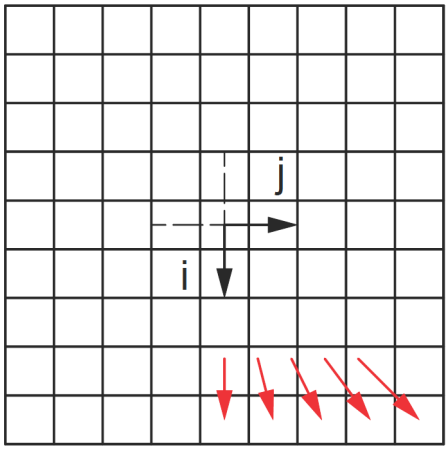



Figure 14: 5 possible directions south-east

# 8 References

## References


[HKM97] Helbing, Dirk ; Keltsch, Joachim ; Molnar, Peter: Modelling the evolution of human trail systems. In: *Nature* 388 (1997), Juli, Nr. 6637, S. 47–50. – ISSN 0028–0836

[HSKM97] Helbing, Dirk ; Schweitzer, Frank ; Keltsch, Joachim ; Molnár, Péter: Active walker model for the formation of human and animal trail systems. In: *Phys. Rev. E* 56 (1997), Sep, Nr. 3, S. 2527–2539. http://dx.doi.org/10.1103/PhysRevE.56.2527. – DOI 10.1103/PhysRevE.56.2527

[PN10] Pfefferle, Jonas ; Nicholas, Pleschke: Simulation of Human Trail Systems / ETH Zurich. 2010. – Forschungsbericht

# A Code

## A.1 Main loops

**Listing 1: simulate.m**

```
clear; clc; hold off;
%% Constants %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
steps = 600;
people = 3;            % number of pedestirans on field
cr_steps= 3;            % number of steps until the next pedestrian will be created
INT=0.2;                % Intensity  (0 < int < 1)  --> good: 0.3
VIS=5.8;                % Visibilty  -> good: 1.8
DUR=350;                % Durability (dur > 1)
Gmax = 1;               % maximal Comfort of walking
importance = 0.18;      % average Importance of destination

%% Examples %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --------- EXAMPLE 1 - Trails of Irchelpark ---------------------------
PATH = imread('path50.bmp', 'bmp');         % PATH
PATH = PATH./255;
PATH = ~PATH;                               % make logical (1=path, 0=no path)
% --> Random entry and destination
% ENTRY = [1,23; 50,17; 12,50; 50,43];
% DEST =  [1,24; 50,20; 14,50; 50,45];                  % Destinations of the
    Pedestrians

% --> possible pathways considering geology of irchelpark
% DEST = [1 23 50 20 ;
%         1 23 50 45 ;
%         50 17 1 24;
%         50 17 14 50;
%         50 43 1 24;
%         12 50 50 20];
%
% ENTRY = [0];

% --------- EXAMPLE 2 - Square  ----------------------------------
PATH = zeros(50,50);
ENTRY = [1,65;1,75;150,49;150,57;150,126;150,136;31,150;42,150];       % Entry Points
    of the Pedestrians
ENTRY = [1,1; 50,49; 1,50; 50,1];
DEST=[1,2; 50,50; 1,49; 49,1];                          % Destinations of

% --------- EXAMPLE 2 - Triangle  ----------------------------------
%  PATH = zeros(50,50);
%  ENTRY = [1,24; 49,1; 49,50];
%  DEST=[1,26; 50,2; 50,49];

n = size(PATH,2);
m = size(PATH,1);
G0 = 0.1*ones(m,n);                         % Initial Ground (set to 0.2 8.5.1011,
    Marius)
G0 = (G0.*~PATH)+(PATH.*Gmax);              % Place the Path onto the ground
```

```
%% Importance of Destination
% Create Importance Matrix (Importance higher close to destinations)
IMP=ones(m,n)*importance;       %Importance of destination (see Pfefferle, Pleschko
    4.1)
% -------------------------------------------------------------
% OPTIONAL: increase importance of destination close to a destination
glocke = zeros(n,m);
for k = 1:size(DEST,1)
    i = DEST(k,1);
    j = DEST(k,2);
    [A,B]= meshgrid(([1:m]-j).^2,([1:n]-i).^2);
    tempglocke = -sqrt(A+B);
    tempglocke = exp(-abs(tempglocke)/3);          % 5
    glocke = glocke + tempglocke;
end
IMP = IMP + (1 - importance)*glocke;
% -------------------------------------------------------------

% INITIAL STATES
G = G0;                                     % Initial ground
P = zeros(n,m);                             % Initial positions of pedestrians
V = updPot(G,VIS);                          % Initial walking potential
step_count = 0;                               % counts the steps from beginning

%% CREATE LOOP  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create and move the first person -> Build PT (Pedestrian Hash-Table)
pid=1;
PT=zeros(1,4);
[P PT(1,:)]=createPedFcn(ENTRY,DEST,P);

for d=1:cr_steps
    [ P PT ]= updPedestrian(V,G,P,PT,pid,VIS,IMP);  %do cr_step times move untitl next
         gerneration,
    [ G ] = updGround(G,P,G0,Gmax,PATH,DUR,INT);        %so peds aren't too closed
    plotGround(V,G,PT,step_count);
    step_count=step_count+1;
end

for i=2:people
    [P PTi]=createPedFcn(ENTRY,DEST,P);
    PT=[PT;PTi];
    pid=size(PT,1);     %number of pedestrians on field

    for d=1:cr_steps        %do cr_steps ntil next creation
        for ii=1:pid        %move loop
            [ P PT ] = updPedestrian(V,G,P,PT,ii,VIS,IMP);
            [ P PT ] = checkArrival(ENTRY,DEST,P,PT,ii);
        end
        G = updGround(G,P,G0,Gmax,PATH,DUR,INT);
        plotGround(V,G,PT,step_count);
        V = updPot(G,VIS);
        step_count=step_count+1;
    end

end

%% UPDATE LOOP  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:steps
```

```
104     % UPDATE Pedestrian
105     for pid = 1:people
106         [ P PT ] = updPedestrian(V,G,P,PT,pid,VIS,IMP);
107         [ P PT ] = checkArrival(ENTRY,DEST,P,PT,pid);
108     end

110     % UPDATE Ground
111     G = updGround(G,P,G0,Gmax,PATH,DUR,INT);
112     % UPDATE Potential every second loop
113     if mod(i,2)==0
114         V = updPot(G,VIS);
115     end
116     % PLOT Ground and Pedestrians
117     plotGround(V,G,PT,step_count);
118     step_count=step_count+1;
119 end
```

## A.2 Pedestrian handling: creation and repositioning

**Listing 2: createPedFcn.m**

```
1  function [ P ped_ENTRY_DEST ] = createPedFcn( ENTRY,DEST,P )
2  %This Function creates a new pedestrian (entry and destination)
3  if(ENTRY ==0)

5      k = size(DEST,1);
6      n = ceil(k*rand());
7      ped_ENTRY_DEST = DEST(n,:);
8      P(DEST(n,1),DEST(n,2)) = 1;

10 else

12     ke=size(ENTRY,1);   %number of entry points
13     kd=size(DEST,1);    %number of destinations

15                     %index can't be 0
16     rndENT = ceil(rand()*ke);
17     rndDES=rndENT;

19     while rndENT==rndDES
20         rndDES=ceil(kd*rand());      %destinastion can't be the entry point
21     end

23     % [ x-entry y-entry x-destination y-destination ]
24     ped_ENTRY_DEST=[ENTRY(rndENT,:),DEST(rndDES,:)];

26     i=ped_ENTRY_DEST(1,1);
27     j=ped_ENTRY_DEST(1,2);

29     P(i,j)=1;

31 end
```

**Listing 3: updPedestrian.m**

```
function [ P PT ] = updPedestrian(V,G,P,PT,pid,VIS,IMP)
    %% P = WALKINGDIRFUNC(G,P,PT,O,PID,VIS,IMP)
    %   G     : Current state of the ground
    %   P     : Current positions of pedestrians
    %   PT    : Attributes of all pedestrians by id
    %   PID   : Current pedestrian
    %   VIS   : Visabilty Matrix                              (m-n-Matrix)
    %   IMP   : Importance

% Ground size
[m n] = size(G);

% Current Position
i = PT(pid,1);
j = PT(pid,2);
curpos = PT(pid,1:2);

%% Resulting Direction
% Relative direction to maximum Potential
W = V;
W(i,j)=0;
% ------------------------------------------------------------
% OPTIONAL: evaluate maxpot in k-sized neighborhood
    k = 25;
    mask = W.*0;
    mask(max(i-k,1):min(n,i+k),max(j-k,1):min(n,j+k)) = ones();
    W = W.*mask;
% ------------------------------------------------------------

[ maxpot x ] = max(W);
[ maxpot y ] = max(maxpot);

maxpotpos = [x(y) y];       % Absolulte position of maximal potential
maxpotdir = (maxpotpos - curpos)/norm(maxpotpos - curpos);

destpos = PT(pid,3:4);      % Absolute Position of destination
destdir = (destpos - curpos)/norm(destpos - curpos);

 I = IMP;
% ------------------------------------------------------------
% OPTIONAL: make importance of destination depend on the angle between the
% maximum Potential an the destination. big angle -> high importance, small
% angle -> low importance
%   --- v.1 ------------
%  impangle = (maxpotdir(2)-destdir(2) >= 0 )* acos( (dot(destdir,maxpotdir)) /(norm(
    destdir)*norm(maxpotdir))) +2*pi*(-(maxpotdir(2)-destdir(2)) > 0);
%  impangle = sin(impangle/2);
%   --- v.2 ------------
% impangle = acos( (dot(destdir,maxpotdir)) /(norm(destdir)*norm(maxpotdir)));
% impangle = impangle/pi;
%
%   I = IMP./IMP * impangle;
% ------------------------------------------------------------


% Resulting walking direction
% IMP = 0 --> 50/50       IMP = 1 --> 0/100
```

```
57  dir = ((.5-.5*I(i,j))*maxpotdir + (.5+.5*I(i,j))*destdir) ...
58      / norm(((.5-.5*I(i,j))*maxpotdir + (.5+.5*I(i,j))*destdir));

60  %% New Position
61  % Angle between resulting walking direction and (1,0).
62  %   (1,0) -> 0, (0,1) -> pi/2, (-1,1) -> pi, etc.
63  angle = sign(dir(2))* acos( (dot(dir,[1 0])) /(norm(dir)*norm([1 0]))) +2*pi*(-dir(2)
        > 0);

65  % Determine which sector is going to be our new position based on the resulting angle
66  [x y newpos] = getNewPos(i,j,m,n,angle);

68  %% Avoid places where other Pedestrians are
69  uu=0;
70  origangle = angle;
71  while ((P(x,y) == 1) || max(max(newpos)) == 0) && uu<4
72      angle = origangle-round(uu)*pi/4; %*cos(2*uu*pi);
73      [x y newpos] = getNewPos(i,j,m,n,angle);
74      uu=uu+0.5;
75  end

77  if sum(sum(newpos)) == 0
78      fprintf('------------------- ERRROORRR -------------\n');
79      fprintf('  A pedestrian does not know where to go\n');
80      PT(pid,:)
81      maxpotdir
82      destdir
83      dir
84      origangle
85      angle
86      fprintf('-----------------------------------------\n');
87      error('No Position found');
88  end

90  P(i,j) = 0;                 % erase the old position
91  P = P + newpos;             % add new position
92  PT(pid,1:2) = [x y];      % update Pedestrian table
```

**Listing 4: getNewPos.m**

```
1   function [x y newpos] = getNewPos(i,j,m,n,angle)

3   sector = mod(floor((angle+pi/8)/(pi/4)),8)+1;
4   S = [ 6 5 4 ;           % The 8 possible sectors.
5         7 0 3 ;
6         8 1 2 ] ;

8   bound = zeros(m+1,n+1);
9   bound(i:i+2,j:j+2) = S;
10  bound = bound(2:m+1,2:n+1);

12  newpos = (bound == sector);
13  newpos(i,j) = 0;

15  [ maxnew x ] = max(newpos);
16  [ foo y ] = max(maxnew);

18  x = x(y);
```

**Listing 5: checkArrival.m**

```
function [P PT] = checkArrival(ENTRY,DEST,P,PT,pid)
    if (PT(pid,1:2)==PT(pid,3:4))    %create new ped if one is at dest
        fprintf('ich bin am ziel %2.0f \n',pid);
        P(PT(pid,1),PT(pid,2))=0;
        [P PTi]=createPedFcn(ENTRY,DEST,P);
        PT(pid,:) = PTi;
    end
```

## A.3 Environmental updates

**Listing 6: updPot.m**

```
function V = updPot(G,VIS)

%% Potential
% Generate walkpot base function
 m = size(G,1);
 n = size(G,2);

 V = zeros(m,n);

 for i = 1:m
     for j = 1:n
        [A,B]=meshgrid(([1:m]-j).^2,([1:n]-i).^2);
        walkpot = -sqrt(A+B);
        walkpot=exp(walkpot./VIS);
        walkpot=walkpot.*G;
        v = sum(sum(walkpot))/(m*n);
        V(i,j) = v;
     end
 end
```

(The lines 12-13 are taken from (PN10))

**Listing 7: updGround.m**

```
function [ Ground ] = updGround(G,P,G0,Gmax,path,dur,int)
    % UPDGROUND(GROUND, G0, GMAX, DUR, VIS, INT)
    % Updates the ground Matrix which models the comfort of walking
    %   G     : Current state of the ground
    %   P     : Current positions of pedestrians
    %   G0    : Initial State of the ground
    %   GMAX  : Maximum comfort
    %   PATH  : PATH Matrix
    %   DUR   : Durability at each point
    %   INT   : Intensity of footprints

restoration = (1/dur).*(G0-G);          % Selfrestoration of the groud
damage = (int*(1-G./Gmax)).*P;          % Damage by footprints
update = (restoration + damage).*~path; % only update where no path is located
```

```
15  Ground = G + update;
```