Modeling Civil Violence

Sam Mundy and Tim Oberhauser

May 26, 2010

Declaration of Authorship

We hereby declare that this report is completely produced by the two of us, we did not use any sources but the ones listed as references. Also this project has been made completely and only for this lecture.

Sam Mundy

Tim Oberhauser

Agreement for Free Download

We hereby agree to make our source code of 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.

Sam Mundy

Tim Oberhauser

Individual Contributions

During the hole working process we worked as a team. Each function was first written as a raw version by one of us and then checked and later on corrected and advanced by both of us. The most important functions such as simulation and initialization were developed together because they also specify the framework.

`activation.m`	Sam
`arrest.m`	Tim
`arrest2.m`	Tim
`birth.m`	Sam
`cop_movement.m`	both
`cop_control.m`	Sam
`death.m`	Tim
`initialisation.m`	both
`kill.m`	Sam
`move.m`	Sam
`movement.m`	Tim
`movement2.m`	both
`movement3.m`	both
`release.m`	Tim
`remove.m`	Sam
`Simulation.m`	both

Table 1: program name and author of the raw versions

Contents

1 Abstract **6**

2 Introduction and Motivation **7**

3 Description of the Model **8**
3.1 Model I by J. M. Epstein 8
3.2 Model II by J. M. Epstein 9
3.3 Extension 9

4 Implementation **11**
4.1 Overview of the Program 11
4.2 Global Variables 12
4.2.1 Field 12
4.2.2 Prison 12
4.3 Main Functions 12
4.3.1 initialization.m 12
4.3.2 activation.m 14
4.3.3 kill.m 14
4.3.4 arrest.m 14
4.3.5 arrest2.m 14
4.3.6 release.m 14
4.3.7 movement.m 15
4.3.8 movement2.m 15
4.3.9 movement3.m 15
4.3.10 cop_movement.m 15
4.3.11 cop_control.m 16
4.3.12 birth.m 16
4.3.13 death.m 16
4.4 Help Functions 17
4.4.1 move.m 17
4.4.2 remove.m 17
4.5 Parameters 17

5 Results and Discussion **18**
5.1 Peaceful Coexistence 18
5.1.1 Stagnation 18
5.1.2 Overpopulation 20
5.2 Ethnic Cleansing 20
5.3 Cop Control 21
5.3.1 Sudden Withdrawal of All Cops 21
5.3.2 Cop Reduction 23
5.3.3 Cop Increase 23

6 Summary **25**

A MATLAB Code **27**

Chapter 1

Abstract

This project is about modeling and simulating civil violence between two groups. In our scenario peacekeepers try to suppress conflicts between members of both groups which try to kill each other. An agent based computational model was therefore used to describe the behaviour of the individuals. Starting with model II by J. M. Epstein we took over and developed rules which define and control the actions of each individual.

We mainly focused on improving the movement rule for the civilians. Instead of moving to a new randomly selected position within vision they should base their choice on safety reasons. The effectiveness of this rule could be shown by applying it to only one group. Members of this group formed clusters and in most scenarios were a lot stronger than the other group, meaning that they survived in larger numbers.

Different scenarios were simulated and many expected outcomes could be observed. Peaceful coexistence prevailed even with no peacekeepers by increasing each groups perception of the others right to exist. Reducing this value without adding peacekeepers led to the extinction of one group. On the other hand violence was easily controllable with enough cops available.

Chapter 2

Introduction and Motivation

The aim of this project is to model the behaviour of two competing groups. A base frame is given by model II by J. M. Epstein [1]. We had to extend the rules of behaviour given in the paper with some own assumptions, because model II was not defined precisely enough.

At first we implemented the model as given by Epstein to get an extendable simulation program. During the programming process, we came up with some ideas to create a more realistic scenario. We mainly focused on improving the movement rule and wanted to study the effect of directed instead of random movement.

Chapter 3

Description of the Model

3.1 Model I by J. M. Epstein

Epstein's model I consists of two agent groups: civilians and cops. The scenario takes place on a 2-D field. The main decision of a civilian in this model is whether to turn active or not. Becoming active means revolting against the government, but also accepting the possibility of being arrested by a cop. The decision is made from certain factors.

Hardship H represents the physical or economic condition of an agent. An agent is more likely to turn against the regime if he is physically fit.

The legitimacy L describes the trust of the people in their authority. Revolution is more probable if the people can't rely on their leaders.

The agent's grievance G is then calculated using the following formula:

$$G = H(1 - L) \tag{3.1}$$

An individual in this model is naturally risk averse. This attribute is specified by R. Hardship H, legitimacy L and risk aversion R are represented by a value between zero and one. Hardship and risk aversion are heterogenous attributes but assumed to be uniformly distributed across all civilians.

Before revolting, an agent has to consider the likelihood of being arrested. Within one's view the cops and the active agents are counted and a cop to active ratio $(C/A)_v$ is calculated. The arrest probability P is then:

$$P = 1 - exp(-k(C/A)_v) \tag{3.2}$$

k is set to a value to get a probability of about $P = 0.9$ for $C = A = 1$, so $k = ln(10) \approx 2.3$.

$P \cdot R$ is called net risk N.

If $G - N$ is greater than a certain threshold T, the agent becomes active or stays active. In case of undercutting the threshold, the civilian either stays quiet or becomes inactive.

3.2 Model II by J. M. Epstein

The second model by Epstein, also the model we implemented, describes the coexistence of two groups A and B, also controlled by cops. Of course now the main concern of one individual is not the government itself but the other group's right to exist, which is represented by the legitimacy. This time there are two legitimacy values, one for each group. The grievances are calculated as follows

$$G_A = H_A(1 - L_B) \text{ and } G_B = H_B(1 - L_A) \tag{3.3}$$

With two present groups, the decision of turning active is also dependent on the other group. The additional condition to turn active is that at least one agent of the other group is present within one's view.

In contrast to model I, after becoming active, actions have to follow, one agent of the other group has to be killed. If there is none within one's grasp, nobody is being killed. Either way the agent becomes inactive again, an agent can therefore only be active for one time step.

Killing another individual makes an agent a criminal. In this model cops do not arrest active agents but only criminals. After a certain time the crime expires, cops are not interested in arresting the agent anymore. We called the expiration time killing term.

The fact that an agent can only be active for a short term led us to the notion that it might make more sense to use a cop to criminal ratio $(C/Cr)_v$ instead of the cop to active ratio $(C/A)_v$. P is in our version derived by

$$P = 1 - exp(-k(C/Cr)_v) \tag{3.4}$$

A Cop decides randomly which criminal within his reach he arrests. Arrested agents are stored in prison for a prefixed number of time steps. The time already served is being counted. Prisoners who have served their time in prison are released.

Another thing becomes necessary in this model, namely aging, birth and death. Without these, the two groups would just wipe each other out. We implemented the aging process by having a uniform age distribution at the beginning and counting the life time afterwards. When a certain maximum age is reached the agent dies. Birth is simply achieved by creating a new agent and setting his age to zero. The number of births is determined by the number of agents of the same group times the prefixed birth rate divided by the maximum age.

3.3 Extension

Epstein let only one agent move at each time step, we gave the movement way more attention and therefore more importance as we increased the number of movements to a certain fraction of the total agents including cops.

The moving agents are randomly picked and furthermore, the step they take is randomly chosen from the free space within their reach.

Our approach to extend the model was to have a more intelligent movement. For instance a criminal agent would not move towards a cop, an agent with high grievance would rather approach the other group, and for all agents it makes sense to move towards agents of the same group.

Another improvement we actualized was, that a cop will arrest the criminal with the highest criminal level within the cop's reach not a randomly chosen one. An agent has a higher criminal level if he has killed another one shortly before or even killed multiple agents at the time.

Chapter 4

Implementation

4.1 Overview of the Program

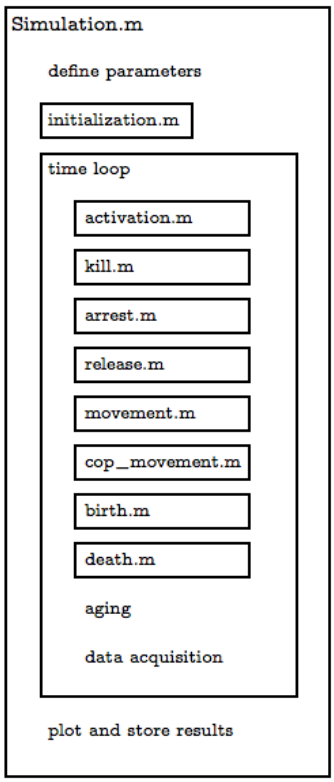

Figure 4.1: program structure

To implement our model in MATLAB we chose to use the following program structure. `Simulation.m` is the main program which is used to define all variable parameters, to run the entire simulation and finally plot and store the results. The simulation is run by first initializing the main variables and the following time loop in which all actions defined by the model are executed sequentially. The main variables are the `field` struct and the `prison` struct. They are created by the function `initialization.m`. During the time loop the actions are simulated by the following functions in the represented order:

First the agents decision to become active and kill is calculated by `activation.m` and executed by `kill.m`. Arrestments by cops are simulated by `arrest.m` or `arrest2.m`. Prisoners are released by calling `release.m`. The movement of Agents is controlled by `movement.m`, `movement2.m` or `movement3.m` and `cop_movement.m` lets cops move. Births and natural deaths are simulated by `birth.m` and `death.m`. `Simulation.m` then saves the `field` for later visualization, collects statistical data and lets agents age before the loop is closed.

This program structure is represented in figure 4.1. `Simulation.m` executes the represented actions or functions.

4.2 Global Variables

The main variables holding all current attributes, states and positions of each agent and cop are `field` and `prison` from the type struct. The current number of agents A, agents B and cops are stored in global variables `numA`, `numB` and `numC` respectively. The size of the field is stored globally in `sizex` and `sizey`. We also defined a global ID count as `IDA`, `IDB` or `IDC` respectively.

4.2.1 Field

The field is implemented as a 1-by-1 struct containing three 1-by-1 substructs: `agentA`, `agentB`, and `cop`. Each one of these substructs includes all attributes and states as m-by-n matrices, where m and n represent height and width of the `field`. The states of agents are `position`, `active` and `criminal` and the attributes are `ID`, `age`, `hardship`, `grievance` and `risk_aversion`, whereas cops only have the state position and the attribute ID. With these m-by-n matrices we store all the current information about every agent and cop on the field. The states and attributes of an agent or cop located at (i,j) is stored in cell (i,j) of the corresponding matrix. This structure is represented in figure 4.2.

The value for position can either be 1 if a person of the corresponding substruct is located in that cell, or 0.

For example: Agent A with the ID k located in cell (i,j) is represented by `field.agentA.position(i,j)=1` and `field.agentA.ID(i,j)=k`. His risk aversion can be found in `field.agentA.risk_aversion(i,j)`.

4.2.2 Prison

The prison consists of the same structure as the field except there is no cop substruct and the states and positions are vectors from the length of the current number of prisoners in the corresponding substruct. Furthermore the states active and criminal do not exist in the prison struct since they are set to 0 once an agent is released. We additionally use `jail_time` as another attribute to save each prisoner's detention time.

4.3 Main Functions

4.3.1 initialization.m

`initialization(legitimacyA,legitimacyB,age_max)` creates an empty `field` and `prison` struct and randomly positions a given number `numA`, `numB`

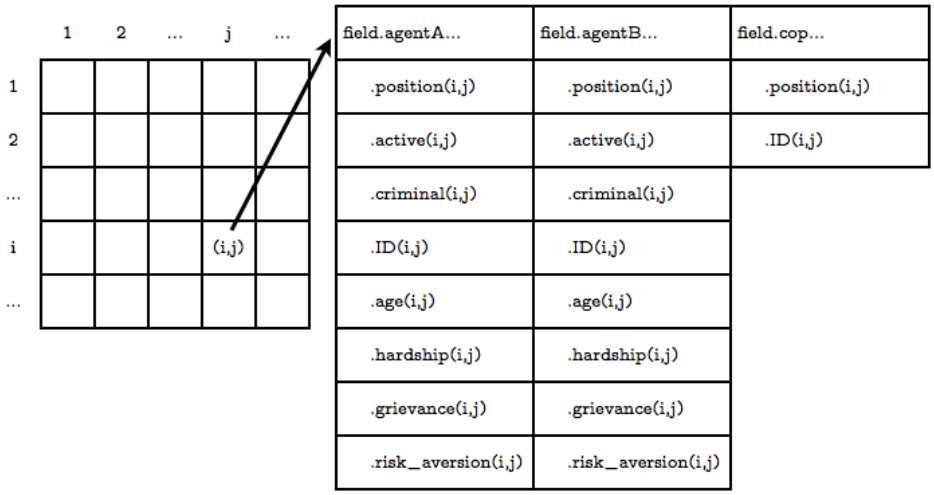

Figure 4.2: field structure

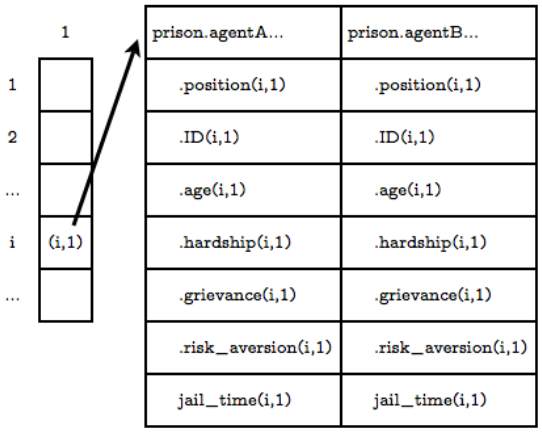

Figure 4.3: prison structure

and `numC` of agents A, agents B and Cops on it. The values for `hardship` and `risk_aversion` are calculated randomly after a uniform distribution on the interval from 0 to 1. The age is also distributed uniformly on the interval from 0 to `age_max`. Every agents `grievance` is then calculated from its `legitimacy` and his own `hardship`.

4.3.2 activation.m

`activation(vision,plausibility,threshold)` determines wether to become active or not for every agent on the `field`. It first all agents A with at least one enemy within their `vision`. For each one of them the `net_risk` is calculated. The same is done for agents B. The function now checks for each agent whether the condition of becoming active or staying quiescent is fulfilled. If $G - N > T$ he becomes active and the state `active` is set to 1 otherwise he stays quiescent and the value 0 is assigned to `active`.

4.3.3 kill.m

`[killcountA,killcountB] = kill(vision,killing_term)` lets all active agents kill an enemy agent if at least one is in sight. The order by which agents kill is random. The total number of killings for both groups A and B are then returned in `killcountA` and `killcountB` respectively.

4.3.4 arrest.m

`arrest(vision)` We assumed that each cop arrests one criminal within his reach if there is at least one. The program checks the `vision` field of every cop and moves one randomly chosen criminal to the `prison`.

4.3.5 arrest2.m

`arrest2(vision)` Similar to the previous version of the arrest function, this function is meant to describe the behaviour of cops, but this time the criminal within the reach of the cop is not randomly chosen, what's more, the agent with the highest criminal level gets arrested. For a small `vision` this will only slightly affect the results because mostly there is only one criminal present. On the other hand if the `vision` is wider it is more likely that extremely aggressive agents are taken off the streets.

4.3.6 release.m

`release(jail_time_max)` If a prisoner has served his time in jail he has to be released. From all the free spaces on the field one is selected and the prisoner is placed, if it's the case that there are none available, the prisoner has to stay at least one more time step until he is free.

4.3.7 movement.m

`movement(mobility,vision)` As already mentioned, in our interpretation of the model movement is much more important than in Epstein's. We prefix a certain fraction of all agents that is able to move, we call this fraction `mobility`. The number of movements is then received by the number of agents multiplied by `mobility`. Now the agents that are able to move are chosen. Every one of them takes one step in one random direction within the agents `vision`.

4.3.8 movement2.m

`movement2(mobility,vision)` Our main idea to extend the model was to have a more intelligent movement. The number of movements stays the same. But now every free position within one's sight is checked whether it is better than the current location. The quality of a location increases with the number of agents of the own group in the view field of the according location. But not only the own group influences the decision where to move, also cops and the other group have to be considered: A criminal will rather not approach a group of cops, therefore the number of cops decreases the quality of a position, if the agent is a criminal. An individual with a high grievance is more willing to approach the other group.

Not only the number of friends, enemies and cops around a position define the quality of it, but also the distances of the agents to it matter. To also consider the distance, we defined a weighting matrix (figure 4.4), which works as follows: For every free position a vision map is generated. This map contains values for each cell of the vision field, the value is 1 if an agent of the same group is located in a cell, G if an agent of the other group is located in a cell or minus the criminal level if a cop is located in a cell. This map is then element wise multiplied by the weighting matrix, to give more importance to the agents located within a smaller radius. The weighting matrix is a two dimensional gaussian distribution with standard variance equals `vision`, and the center at the checked position.

4.3.9 movement3.m

`movement3(mobility,vision)` only group A moves intelligently - the other one randomly.

4.3.10 cop_movement.m

`cop_movement(mobility,vision)` moves a number of randomly chosen cops each to a new randomly determined position within his `vision`. The number of movements is determined as in `movement.m`.

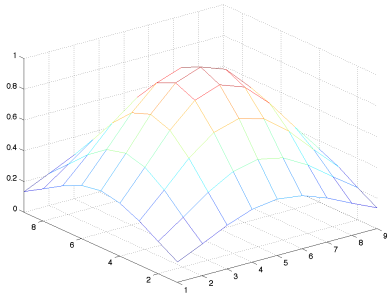

Figure 4.4: mesh plot of the weighting matrix for `vision`=4

4.3.11 cop_control.m

`cop_control(cop_difference)` increases or reduces the number of cops by the value of `cop_difference` by placing new cops at random free positions on the field or removing randomly selected cops.

4.3.12 birth.m

`birth(birth_rate,age_max)` chooses agents on the field that will be duplicated to a new randomly chosen position as long as there is enough free space. The agents are chosen by a birth probability which is the ratio of `birth_rate` to `age_max`. By duplicating all attributes are copied except for age which is set to 0.

4.3.13 death.m

`death(age_max)` removes all agents whose age exceeds `age_max`.

4.4 Help Functions

4.4.1 move.m

`move(type,from,x0,y0,to,x,y)` duplicates an agent or cop defined by `type` to a new position `(y,x)` on the `field` or `prison` which can be chosen by the string `to`.

4.4.2 remove.m

`remove(type,x,y)` deletes an agent or cop specified by `type` from the position `(y,x)` on the `field`.

4.5 Parameters

variable name	**description**
`sizex, sizey`	field dimensions
`t_max`	number of iterations
`numA, numB, numC`	initial agent numbers
`legitimacyA, legitimacyB`	L, the other groups right to exist
`vision`	visible range
`plausibility`	k, plausibility
`threshold`	T, threshold
`jail_time_max`	standard detention time
`age_max`	standard life time
`birth_rate`	birth rate
`mobility`	ratio of moving to total number of agents
`killing_term`	expiration time of a crime

Table 4.1: parameters

Chapter 5

Results and Discussion

To be able to compare results from different setups, some parameters have to remain the same for all experiments. Parameters we did not change - if not else declared - to obtain the following results are shown in table 5.1.

field size	50 x 50
`mobility`	0.1
`vision`	4
k	$ln(10)$
T	0.1

Table 5.1: fixed parameters for all simulation runs

5.1 Peaceful Coexistence

5.1.1 Stagnation

To test each version we tried to reproduce a state of peaceful coexistence. This state is reached by setting both legitimacy values to 1. Recalling definition of grievance (3.3), grievance always turns out zero, resulting in no agent being able to turn active and killing anyone.

With both legitimacies set above 0.9, no killings emerge and therefore no cops are needed to control the field. To hold the population constant the birthrate has to be fixed at 1. The resulting development can be taken from figure 5.1 The number of agents on the field can be adjusted arbitrarily. We chose `numA=numB=1000`. Running with two different numbers, we could also show that both groups develop independently.

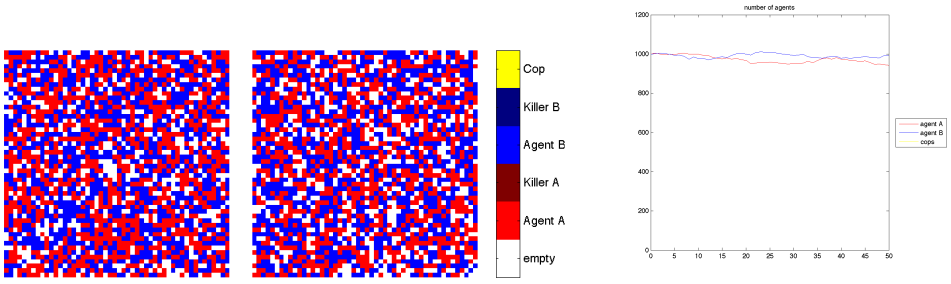

Figure 5.1: field at t=0 and t=t_max and population for stagnation using `movement.m`

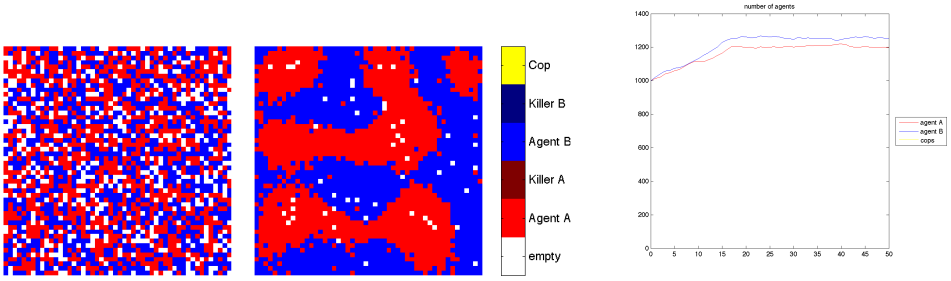

Figure 5.2: field at t=0 and t=t_max and population for overpopulation using `movement2.m`

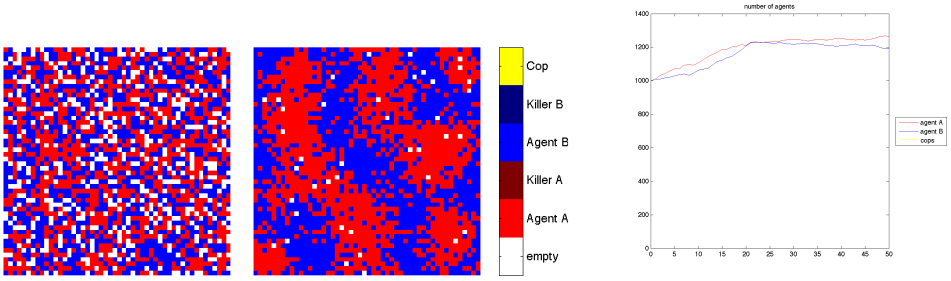

Figure 5.3: field at t=0 and t=t_max and population for overpopulation using `movement3.m`

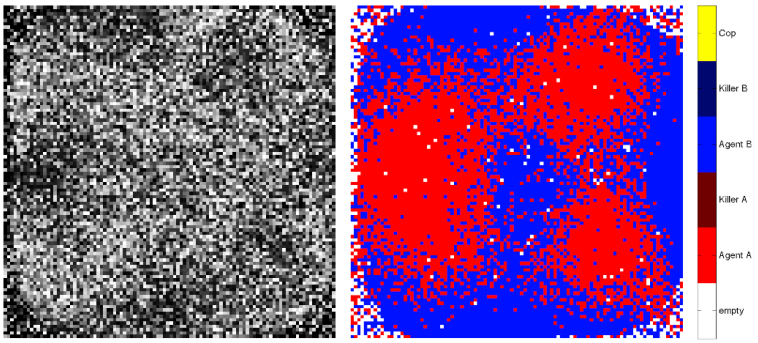

Figure 5.4: grievance map (black representing empty spaces or the lowest possible grievance and white representing high grievance) and field, both at t=t_max

5.1.2 Overpopulation

Another interesting development is to see how agents arrange when overpopulation is reached. The number of births is expected to collapse as soon as the state of overpopulation is reached.

Overpopulation can be achieved by setting the legitimacy of both groups to 1 and fixing the birth rate at a higher value. We used a birth rate of 1.5 to produce overpopulation.

To compare the two directed movement fuctions we ran one simulation using `movement2.m` and one using `movement3.m`. The resulting field and population course are shown in figures 5.2 and 5.3.

Based on our movement rule, agents with higher grievance are more likely to move towards agents of the other group. To validate this, we set both legitimacy values to 0.8, therefore the grievance is ≥ 0. To still ensure peaceful coexistence we set T to 1, making it impossible for an agent to become active. To improve the quality of the results this time we used field dimensions of 100x100 and only 30 time steps, to still have vivid movement, which could be inhibited after reaching overpopulation. A `vision` of 20 made sense to us because of the greater field size. If we implemented our movement rule right, agents with higher grievance would surround a posse. The resulting grievance map is compared to the field in figure 5.4.

5.2 Ethnic Cleansing

To observe actual violence, both legitimacy values have to be decreased to values around 0.8. Ethnic cleansing only occurs with the random movement, because agents with directed movement form posses in the beginning and then remain in a more or less steady state.

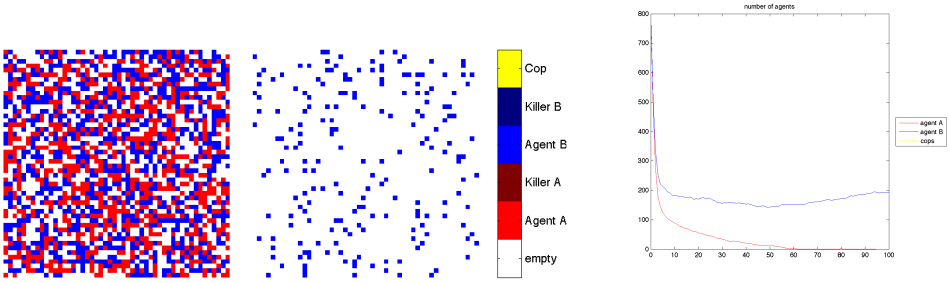

Figure 5.5: field at t=0 and t=t_max and population for ethnic cleansing using `movement.m`

We observed that the development with legitimacy values of 0.8 and a threshold of 0.1. The resulting collapse of the population numbers at the beginning is very abrupt. To observe ethnic cleansing this is not fatal, because the final state is the one of interest.

A birth rate of 1.5 and initial `numA=numB=800` seemed reasonable. The simulation results are shown in figure 5.5.

5.3 Cop Control

5.3.1 Sudden Withdrawal of All Cops

Interesting behaviours can be studied if all cops are withdrawn at a certain time step. Starting from random movement of the agents, the sudden cop removal will only result in ethnic cleansing shifted in time. Much more interesting is to let the agents form posses and and then remove the cops. We decided to run the simulation with initial `numA=numB=1000, numC=200` and legitimacy values of 0.8. This time a threshold of 0.19 made more sense to us, to have a smoother population drop.

We removed all cops at t = 30, because the formation of the agents is then completed sufficiently.

We ran this simulation once with both groups moving intelligently (figure 5.6). This results in an expected way, after the cop removal, the number of killings explodes. Which group forces through at the end is random, a small advantage at the beginning leads to a much higher population at the end.

A second simulation was run with the same parameters, but with only group A moving intelligently and B moving randomly (figure 5.7). Surprisingly group B came out on top But this is not always the case, running the same simulation multiple times A turned out as the winner most of the time.

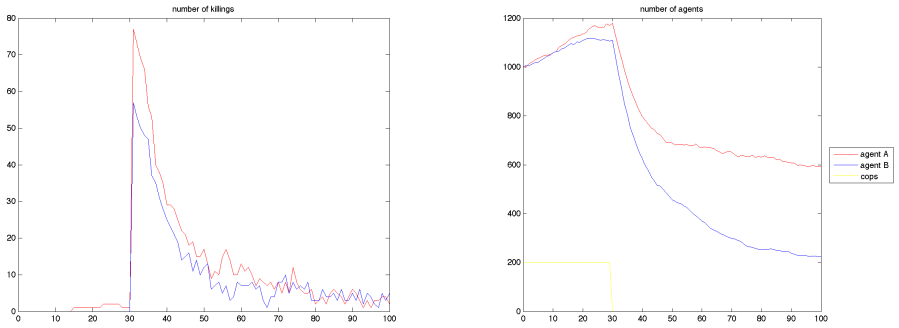

Figure 5.6: field at t=0 and t=t_max and population for cop withdrawal using `movement2.m`

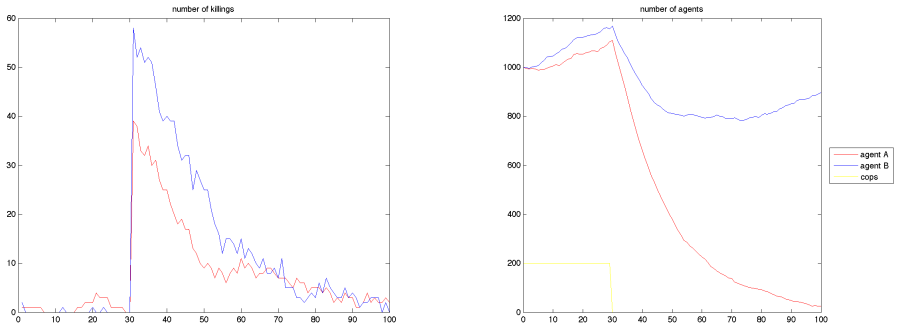

Figure 5.7: field at t=0 and t=t_max and population for cop withdrawal using `movement3.m`

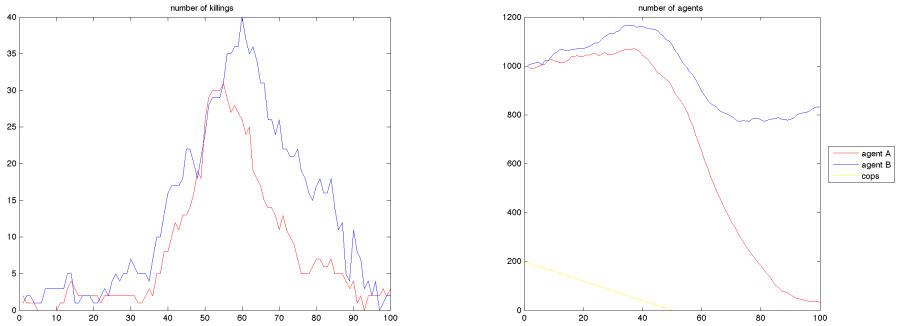

Figure 5.8: killings and population course with a constant cop reduction rate

5.3.2 Cop Reduction

An interesting development takes place when cops are reduced at a constant rate. We again fixed legitimacies at 0.8, birth rate at 1.5, initial civilian numbers at `numA=numB=1000` and used movement.m. As the initial number of cops we chose a value to have more or less peaceful coexistence, `numC=200` worked just fine. The number of cops was reduced at a rate to have no more cops left at t=50, the `cop_difference` for each time step is then -4.

Notable about the course of the killings is that there seems to be a minimal number of cops to preserve a reasonable level of violence (figure 5.8). This is the case at t $\approx$ 30, `numC` then is 80, `numA` $\approx$ `1050` and `numB` $\approx$ `1150`. A cop to total civilians ratio of 0.036 is enough to stabilize the situation at the given conditions.

5.3.3 Cop Increase

A much more important question is the one about how fast cops have to intervene when violence breaks out in crowds. Transferring this to the model, cops have to be increased. We decided to do this again at a constant rate. Thinking of the speed of the killings when ethnic cleansing occurred, cops have to be added at a high rate, 15 cops more each time step worked out well. The number of killings could be reduced very fast, after 8 time iterations a very low level was already reached (figure 5.9, `movement.m` was used). At this time the cop to total civilians ratio is about 0.07. This value is higher than the one to preserve peace with cop reduction, this might be the case because as soon as violence has breaks out, the cop to criminal ratio is much smaller and therefore more violence takes place.

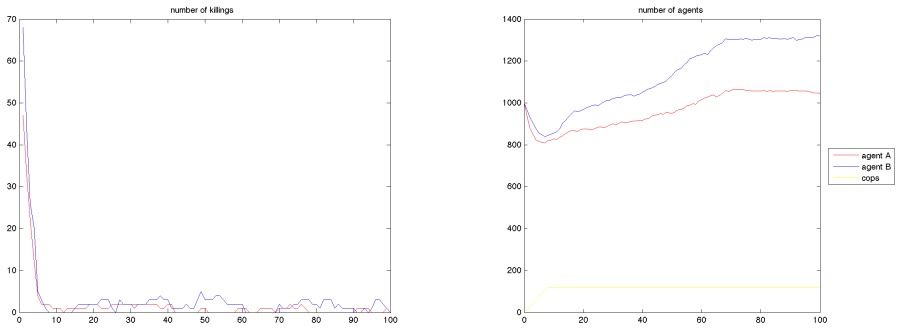

Figure 5.9: killings and population course with a constant cop increasing rate

Chapter 6

Summary

The results obtained from our model are often comparable to simplified human behavior. We were able to produce certain expected effects.

A state of peaceful coexistence even without cops was easily achieved by increasing both legitimacy values to 1. A high perception of the other groups right to exist should not lead to any killings. In this scenario a birth rate above 1 leads to overpopulation. Considering that every civilian has on an average more than one child this is the expected outcome.

The level of violence rises rapidly when the legitimacy is set below $1-T$. Without peacekeepers this ultimately leads to the extinction of one group. Running this scenario with the advanced movement rule applied to only one of the two groups gives them such an advantage that they prevail. It seems like the formation of clusters is an effective protection against enemies. Looking at the individual levels of grievance, we see that weak civilians tend to allocate at the center of clusters leaving stronger ones surrounding them, where strength resembles to grievance. This way the civilians that are more likely to turn active and kill are at the front, where they can reach enemies, which will lead to more killings.

With a rising number of cops though, violence can be controlled quite easily. Under a sufficient number of cops violent coexistence prevails. By varying the number of cops over time we recognized the level of violence is not only dependent on the current state but also on the past. A hysteresis phenomena can be observed.

Bibliography

[1] Epstein J M 2002, *Modeling civil violence: An agent-based computational approach*, PNAS 2002 99: 7243-7250

Appendix A

MATLAB Code

```
% Simulation only runs on MATLAB 2010
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

close all
clear all

% % Set Parameters

% define global variables

global field prison sizex sizey numA numB numC IDA IDB IDC

% field properties

sizex=50;
sizey=50;

% time

t_max=50;
t_vec=1:t_max;

% agent & cop properties

numA=900;
numB=900;
numC=0;

legitimacyA=0.8;
legitimacyB=0.8;

vision=4;
plausability=log(10);
threshold=0.19;

jail_time_max=10;
age_max=t_max;
birth_rate=1.4;
mobility=0.1;

killing_term=10;

track_ID=numA+1;
track=zeros(t_max,1);


% % Initialisation

initialization(legitimacyA,legitimacyB,age_max);

% define variables to capture statistics

numA_vec=zeros(t_max+1,1);
numB_vec=numA_vec;
numC_vec=numA_vec;
numA_vec(1)=numA;
numB_vec(1)=numB;
numC_vec(1)=numC;
IDA=numA+1;
```

```
IDB=numB+1;
IDC=numC+1;

killcount_vecA=zeros(t_max,1);
killcount_vecB=zeros(t_max,1);

birthcount_vecA=zeros(t_max,1);
birthcount_vecB=zeros(t_max,1);

prisoncount_vecA=zeros(t_max+1,1);
prisoncount_vecB=zeros(t_max+1,1);

% set up visualisation

image=zeros(sizey,sizex,1,t_max+1);
image(:,:,1,1)=field.agentA.position+field.agentA.criminal+...
    3*field.agentB.position+field.agentB.criminal+5*field.cop.position;


% % Time-Loop

for t=t_vec

    IDAold=IDA;
    IDBold=IDB;

%     % tracking function
%     [current_tracky current_trackx]=find(field.agentA.ID==track_ID);
%     if isempty(current_tracky)
%         if isempty(find(prison.agentA.ID==track_ID))
%             track(t,1:2)=NaN;
%         else
%             track(t,1:2)=0;
%         end
%     else
%
%         track(t,1:2)=[current_tracky current_trackx];
%         image(track(t,1),track(t,2),t)=6;
%     end

    % actions
    activation(vision,plausability,threshold,killing_term);
    [killcountA,killcountB] = kill(vision,killing_term);
    arrest2(vision);
    release(jail_time_max);
    movement2(mobility,vision);
    cop_movement(mobility,vision);
    birth(birth_rate,age_max);
    death(age_max);

    % cop removal
    if t<=30
        cop_control(3);
    end


    % visualisation
    image(:,:,1,t+1)=field.agentA.position+field.agentA.criminal./...
        (field.agentA.criminal+1)+3*field.agentB.position+...
        field.agentB.criminal./(field.agentB.criminal+1)+...
        5*field.cop.position;

    % aging process
    prison.agentA.jail_time=prison.agentA.jail_time+1;
    prison.agentB.jail_time=prison.agentB.jail_time+1;
```

```
    field.agentA.age=field.agentA.age+field.agentA.position;
    field.agentB.age=field.agentB.age+field.agentB.position;
    [crimy crimx]=find(field.agentA.criminal>0);
    for i=1:length(crimy)
    field.agentA.criminal(crimy(i),crimx(i))=...
        field.agentA.criminal(crimy(i),crimx(i))-1;
    end
    [crimy crimx]=find(field.agentB.criminal>0);
    for i=1:length(crimy)
    field.agentB.criminal(crimy(i),crimx(i))=...
        field.agentB.criminal(crimy(i),crimx(i))-1;
    end

    % display time step
    disp(t);

    % gather stats
    numA_vec(t+1)=numA;
    numB_vec(t+1)=numB;
    numC_vec(t+1)=numC;
    killcount_vecA(t)=killcountA;
    killcount_vecB(t)=killcountB;
    birthcount_vecA(t)=IDA-IDAold;
    birthcount_vecB(t)=IDB-IDBold;
    prisoncount_vecA(t+1)=length(prison.agentA.position);
    prisoncount_vecB(t+1)=length(prison.agentB.position);

end

% % Visualizations

% field visualization over time
figure('Color',[1 1 1])
hold on
%set(gca,'xtick',[],'ytick',[]) % remove x and y ticks around field

axis equal
axis(0.5+[0 sizex 0 sizey]) % set axes
axis off

colormap([1 1 1; 1 0 0; 0.5 0 0; 0 0 1; 0 0 0.5; 1 1 0])
caxis([-0.5 5.5])
colorbar('YTickLabel',...
    {'empty','Agent A','Killer A','Agent B',...
     'Killer B','Cop'},'ytick',[0 1 2 3 4 5]) % colormapzeug
drawnow

for t=[t_vec t_max+1]
    imagesc(image(:,:,1,t))
    drawnow
    pause(0.1)
end


% initial and final field
h=figure('Color',[1 1 1]);
vergleich=[image(:,:,1,1) zeros(sizey,5) image(:,:,1,t_max+1)];
imagesc(vergleich)
colormap([1 1 1; 1 0 0; 0.5 0 0; 0 0 1; 0 0 0.5; 1 1 0])
caxis([-0.5 5.5])
colorbar('YTickLabel',...
    {'empty','Agent A','Killer A','Agent B',...
     'Killer B','Cop'},'ytick',[0 1 2 3 4 5]) % colormapzeug
axis equal
axis(0.5+[0 sizex*2+5 0 sizey]) % set axes
```

```
axis off
print(h,'-dpng',['bericht/vergleichnA' num2str(numA_vec(1))...
   'nB' num2str(numB_vec(1)) 'nC' num2str(numC) 'lA'...
   num2str(legitimacyA) 'lB' num2str(legitimacyB) '.png'])

% % alternate tracking visualization
% figure(1)
% hold on
% %set(gca,'xtick',[],'ytick',[]) % remove x and y ticks around field
%
% axis equal
% axis(0.5+[0 sizex 0 sizey]) % set axes
% axis off
%
% colormap([1 1 1; 1 0 0; 0.5 0 0; 0 0 1; 0 0 0.5; 0 0 0; 0 1 1])
% caxis([-0.5 6.5])
% colorbar('YTickLabel',...
%     {'empty','Agent A','Killer A','Agent B',...
%      'Killer B','Cop','Tracky'},'ytick',[0 1 2 3 4 5 6]) % colormapzeug
% %drawnow
%
% for t=[t_vec t_max+1]
%     imagesc(image(:,:,1,t))
%     drawnow
%     pause(0.15)
% end
% %track end

% % create movie
% mov = immovie(image+1,[1 1 1; 1 0 0; 0.5 0 0; 0 0 1; 0 0 0.5; 1 1 0]);
% movie2avi(mov,'test','fps',10,'compression','none','colormap',...
%     [1 1 1; 1 0 0; 0.5 0 0; 0 0 1; 0 0 0.5; 0 0 0])

scrsz = get(0,'ScreenSize');
figure('Position',[10 50 scrsz(3)*0.8 scrsz(4)-140],'Color',[1 1 1])
subplot(2,2,1)
plot([0 t_vec],numA_vec,'r-',[0 t_vec],numB_vec,'b-',[0 t_vec],...
    numC_vec,'y-')
legend('agent A','agent B','cops','Location','southoutside',...
    'Orientation','horizontal')
title('number of agents')

subplot(2,2,2)
plot([0 t_vec],prisoncount_vecA,'r-',[0 t_vec],prisoncount_vecB,'b-')
title('number of prisoners')

subplot(2,2,3)
plot(t_vec,killcount_vecA,'r-',t_vec,killcount_vecB,'b-')
title('number of killings')

subplot(2,2,4)
plot(t_vec,birthcount_vecA,'r-',t_vec,birthcount_vecB,'b-')
title('number of births')
```

Published with MATLAB® 7.10

```
function [] = initialization(legitimacyA,legitimacyB,age_max)
%INITIALIZATION
%   initialization(legitimacyA,legitimacyB,age_max) creates an empty field
%   and prison struct and randomly positions a given number numA, numB and
%   numC of agents A, agents B and Cops on it. The values for hardship and
%   risk_aversion are calculated randomly after a uniform distribution on
%   the interval from 0 to 1. The age is also distributed uniformly on the
%   interval from 0 to age_max. Every agents grievance is then calculated
%   from their legitimacy and his own hardship.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field prison sizex sizey numA numB numC

% create empty field
field=struct(...
    'agentA',struct(...
        'position',zeros(sizey,sizex),...
        'active',zeros(sizey,sizex),...
        'hardship',zeros(sizey,sizex),...
        'grievance',zeros(sizey,sizex),...
        'criminal',zeros(sizey,sizex),...
        'age',zeros(sizey,sizex),...
        'risk_aversion',zeros(sizey,sizex),...
        'ID',zeros(sizey,sizex)),...
    'agentB',struct(...
        'position',zeros(sizey,sizex),...
        'active',zeros(sizey,sizex),...
        'hardship',zeros(sizey,sizex),...
        'grievance',zeros(sizey,sizey),...
        'criminal',zeros(sizey,sizex),...
        'age',zeros(sizey,sizex),...
        'risk_aversion',zeros(sizey,sizex),...
        'ID',zeros(sizey,sizex)),...
    'cop',struct('position',zeros(sizey,sizex),...
        'ID',zeros(sizey,sizex))...
    );

% create empty prison
prison=struct(...
    'agentA',struct(...
        'position',zeros(0,1),...
        'hardship',zeros(0,1),...
        'grievance',zeros(0,1),...
        'risk_aversion',zeros(0,1),...
        'age',zeros(0,1),...
        'jail_time',zeros(0,1),...
        'ID',zeros(sizey,sizex)),...
    'agentB',struct(...
        'position',zeros(0,1),...
        'hardship',zeros(0,1),...
        'grievance',zeros(0,1),...
        'risk_aversion',zeros(0,1),...
        'age',zeros(0,1),...
        'jail_time',zeros(0,1),...
        'ID',zeros(sizey,sizex)));

% place agents
placement=rand(sizey,sizex);
```

```
for i=1:numA
    [unn,y]=max(placement);
    [~,x]=max(unn);
    field.agentA.position(y(x),x)=1;
    field.agentA.ID(y(x),x)=i;
    placement(y(x),x)=0;
end

for i=1:numB
    [unn,y]=max(placement);
    [~,x]=max(unn);
    field.agentB.position(y(x),x)=1;
    field.agentB.ID(y(x),x)=i;
    placement(y(x),x)=0;
end

% place cops
for i=1:numC
    [unn,y]=max(placement);
    [~,x]=max(unn);
    field.cop.position(y(x),x)=1;
    field.cop.ID(y(x),x)=i;
    placement(y(x),x)=0;
end

% generate hardship
field.agentA.hardship=field.agentA.position.*rand(sizey,sizex);
field.agentB.hardship=field.agentB.position.*rand(sizey,sizex);

% calculate individual grievance
field.agentA.grievance=field.agentA.hardship*(1-legitimacyB);
field.agentB.grievance=field.agentB.hardship*(1-legitimacyA);

% generate risk aversion
field.agentA.risk_aversion=field.agentA.position.*rand(sizey,sizex);
field.agentB.risk_aversion=field.agentB.position.*rand(sizey,sizex);

% generate uniform age distribution
field.agentA.age=randi(age_max,sizey,sizex).*field.agentA.position;
field.agentB.age=randi(age_max,sizey,sizex).*field.agentB.position;

end
```

Published with MATLAB® 7.10

```
function [] = activation(vision,plausability,threshold,killing_term)
%ACTIVATION Activate agents.
%   ACTIVATION(vision,plausability,threshold) determines wether to become
%   active or not for every agent on the field. It first finds all agents A
%   with at least one enemy within their vision. For each one of them the
%   net risk is calculated. The same is done for agents B. The function now
%   checks for each agent wether the condition of becoming active or
%   staying quiescent is fulfilled. If G - N > T he becomes active and
%   the state active is set to 1 otherwise he stays quiescent and the value
%   0 is assigned to active.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field sizex sizey numA numB

% locate agents A
[posy,posx]=find(field.agentA.position);

% check for neighbours
check=zeros(sizey,sizex);

for i=1:numA
    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;
    if posy(i)-vision<=0
        visionymin=posy(i)-1;
    end
    if posy(i)+vision>=sizey+1
        visionymax=sizey-posy(i);
    end
    if posx(i)-vision<=0
        visionxmin=posx(i)-1;
    end
    if posx(i)+vision>=sizex+1
        visionxmax=sizex-posx(i);
    end
    check(posy(i),posx(i))=sum(sum((field.agentB.position(posy(i)-...
        visionymin:posy(i)+visionymax,posx(i)-visionxmin:posx(i)+...
        visionxmax))));
end

[posy,posx]=find(check);

% calculate cop/criminal ratio
ratio=zeros(sizey,sizex);

for i=1:length(posy)
    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;
    if posy(i)-vision<=0
        visionymin=posy(i)-1;
    end
    if posy(i)+vision>=sizey+1
        visionymax=sizey-posy(i);
    end
```

```
    if posx(i)-vision<=0
        visionxmin=posx(i)-1;
    end
    if posx(i)+vision>=sizex+1
        visionxmax=sizex-posx(i);
    end
    ratio(posy(i),posx(i))=sum(sum(field.cop.position(posy(i)-...
        visionymin:posy(i)+visionymax,posx(i)-visionxmin:posx(i)+...
        visionxmax)))/(1+sum(sum(field.agentA.criminal(posy(i)-...
        visionymin:posy(i)+visionymax,posx(i)-visionxmin:posx(i)+...
        visionxmax)))/killing_term+sum(sum((field.agentB.criminal...
        (posy(i)-visionymin:posy(i)+visionymax,posx(i)-...
        visionxmin:posx(i)+visionxmax))))/killing_term);
end

% calculate arrest probability
probability=1-exp(-plausability*ratio);

% calculate net risk
net_risk=probability.*field.agentA.risk_aversion;

Aposy=posy;
Aposx=posx;
net_riskA=net_risk;

% locate agents B
[posy,posx]=find(field.agentB.position);

% check neighbours
check=zeros(sizey,sizex);

for i=1:numB
    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;
    if posy(i)-vision<=0
        visionymin=posy(i)-1;
    end
    if posy(i)+vision>=sizey+1
        visionymax=sizey-posy(i);
    end
    if posx(i)-vision<=0
        visionxmin=posx(i)-1;
    end
    if posx(i)+vision>=sizex+1
        visionxmax=sizex-posx(i);
    end
    check(posy(i),posx(i))=sum(sum((field.agentA.position(posy(i)-...
        visionymin:posy(i)+visionymax,posx(i)-visionxmin:posx(i)+...
        visionxmax))));
end

[posy,posx]=find(check);

% calculate cop/criminal ratio
ratio=zeros(sizey,sizex);

for i=1:length(posy)
    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;
    if posy(i)-vision<=0
        visionymin=posy(i)-1;
```

```
    end
    if posy(i)+vision>=sizey+1
        visionymax=sizey-posy(i);
    end
    if posx(i)-vision<=0
        visionxmin=posx(i)-1;
    end
    if posx(i)+vision>=sizex+1
        visionxmax=sizex-posx(i);
    end
    ratio(posy(i),posx(i))=sum(sum(field.cop.position(posy(i)-...
        visionymin:posy(i)+visionymax,posx(i)-visionxmin:posx(i)+...
        visionxmax)))/(1+sum(sum(field.agentA.criminal(posy(i)-...
        visionymin:posy(i)+visionymax,posx(i)-visionxmin:posx(i)+...
        visionxmax)))/killing_term+sum(sum((field.agentB.criminal...
        (posy(i)-visionymin:posy(i)+visionymax,posx(i)-...
        visionxmin:posx(i)+visionxmax))))/killing_term);
end

% calculate arrest probability
probability=1-exp(-plausability*ratio);

% calculate net risk
net_risk=probability.*field.agentB.risk_aversion;

% Activation

% Agents A
for i=1:length(Aposy)
    if field.agentA.grievance(Aposy(i),Aposx(i))-...
            net_riskA(Aposy(i),Aposx(i))<threshold
        field.agentA.active(Aposy(i),Aposx(i))=0;
    else
        field.agentA.active(Aposy(i),Aposx(i))=1;
    end
end

% Agents B
for i=1:length(posy)
    if field.agentB.grievance(posy(i),posx(i))-...
            net_risk(posy(i),posx(i))<threshold
        field.agentB.active(posy(i),posx(i))=0;
    else
        field.agentB.active(posy(i),posx(i))=1;
    end
end

end
```

Published with MATLAB® 7.10

```
function [killcountA,killcountB] = kill(vision,killing_term)
%KILL Kill enemy agents.
%   [killcountA,killcountB] = kill(vision,killing_term) lets all active
%   agents kill an enemy agent if at least one is in sight. The order by
%   which agents kill is random. The total number of killings for both
%   groups A and B are then returned in killcountA and killcountB
%   respectively.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field sizex sizey

killcountA=0;
killcountB=0;

while sum(sum(field.agentA.active+field.agentB.active))

    [Aposy,Aposx]=find(field.agentA.active);
    [Bposy,Bposx]=find(field.agentB.active);
    Position=[Aposy Aposx; Bposy Bposx];

    % random choose Agent
    n=randi(size(Position,1));

    % check if there are victims
    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;
    if Position(n,1)-vision<=0
        visionymin=Position(n,1)-1;
    end
    if Position(n,1)+vision>=sizey+1
        visionymax=sizey-Position(n,1);
    end
    if Position(n,2)-vision<=0
        visionxmin=Position(n,2)-1;
    end
    if Position(n,2)+vision>=sizex+1
        visionxmax=sizex-Position(n,2);
    end

    if n<=length(Aposy)
        victim=(field.agentB.position(Position(n,1)-...
            visionymin:Position(n,1)+visionymax,Position(n,2)-...
            visionxmin:Position(n,2)+visionxmax));
    else
        victim=(field.agentA.position(Position(n,1)-...
            visionymin:Position(n,1)+visionymax,Position(n,2)-...
            visionxmin:Position(n,2)+visionxmax));
    end

    % kill victim
    if sum(sum(victim))
        [victimy,victimx]=find(victim);
        i=randi(length(victimy));
        if n<=length(Aposy)
            killcountA=killcountA+1;
            remove(Position(n,2)-visionxmin+victimx(i)-1,Position(n,1)-...
```

```
                visionymin+victimy(i)-1,'B');
            field.agentA.criminal(Position(n,1),Position(n,2))=...
                field.agentA.criminal(Position(n,1),Position(n,2))+...
                killing_term;
        else
            killcountB=killcountB+1;
            remove(Position(n,2)-visionxmin+victimx(i)-1,Position(n,1)-...
                visionymin+victimy(i)-1,'A');
            field.agentB.criminal(Position(n,1),Position(n,2))=...
                field.agentB.criminal(Position(n,1),Position(n,2))+...
                killing_term;
        end


    end

    % remove from active
    if n<=length(Aposy)
        field.agentA.active(Position(n,1),Position(n,2))=0;
    else
        field.agentB.active(Position(n,1),Position(n,2))=0;
    end
end

end
```

Published with MATLAB® 7.10

```
function [] = arrest(vision)
%ARREST Arrest agents.
%   ARREST(vision) checks the vision field of every cop and moves one
%   randomly chosen criminal to the prison.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

global field prison sizex sizey numA numB numC

% locate cops
[Cposy,Cposx]=find(field.cop.position);

for i=1:numC
    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;
    if Cposy(i)-vision<=0
        visionymin=Cposy(i)-1;
    end
    if Cposy(i)+vision>=sizey+1
        visionymax=sizey-Cposy(i);
    end
    if Cposx(i)-vision<=0
        visionxmin=Cposx(i)-1;
    end
    if Cposx(i)+vision>=sizex+1
        visionxmax=sizex-Cposx(i);
    end

    % locate criminal agents A within cop vision

    [Aposy,Aposx]=find(field.agentA.criminal(...
        Cposy(i)-visionymin:Cposy(i)+visionymax,...
        Cposx(i)-visionxmin:Cposx(i)+visionxmax));

    Aposy=Aposy-1+Cposy(i)-visionymin;
    Aposx=Aposx-1+Cposx(i)-visionxmin;

    % locate criminal agents B within cop vision

    [Bposy,Bposx]=find(field.agentB.criminal(...
        Cposy(i)-visionymin:Cposy(i)+visionymax,...
        Cposx(i)-visionxmin:Cposx(i)+visionxmax));

    Bposy=Bposy-1+Cposy(i)-visionymin;
    Bposx=Bposx-1+Cposx(i)-visionxmin;

    % arrest one random criminal in each time step, if at least one is in
    % vision
    if length(Aposy)+length(Bposy)>=1

        random=randi(length(Aposy)+length(Bposy));

        if random<=length(Aposy)
            % count enclosed agents A
            prisoncountA=length(prison.agentA.position);

            % place in prison with all attributes
            [field,prison,numA,numB] = move(...
```

```
                field,prison,numA,numB,...
                'A','field',Aposx(random),Aposy(random),...
                'prison',1,prisoncountA+1);

            % remove agent A from field
            [field,numA,numB,numC] = remove(field,numA,numB,numC,...
                Aposx(random),Aposy(random),'A');

        else
            % count enclosed agentBs
            prisoncountB=length(prison.agentB.position);

            % place in prison with all attributes
            [field,prison,numA,numB] = move(...
                field,prison,numA,numB,...
                'B','field',Bposx(random-length(Aposy)),Bposy(random-...
                length(Aposy)),'prison',1,prisoncountB+1);

            % remove agent B from field
            [field,numA,numB,numC] = remove(field,numA,numB,numC,...
                Bposx(random-length(Aposy)),...
                Bposy(random-length(Aposy)),'B');
        end
    end
end

end
```

Published with MATLAB® 7.10

```
function [] = arrest2(vision)
%ARREST Arrest agents.
%   ARREST(vision) checks the vision field of every cop and moves the agent
%   with the highest criminal level to the prison.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field prison sizex sizey numC

% locate cops
[Cposy,Cposx]=find(field.cop.position);

for i=1:numC
    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;
    if Cposy(i)-vision<=0
        visionymin=Cposy(i)-1;
    end
    if Cposy(i)+vision>=sizey+1
        visionymax=sizey-Cposy(i);
    end
    if Cposx(i)-vision<=0
        visionxmin=Cposx(i)-1;
    end
    if Cposx(i)+vision>=sizex+1
        visionxmax=sizex-Cposx(i);
    end


    % arrest the criminal with the highest criminal level
    viewmatrixA=field.agentA.criminal(...
        Cposy(i)-visionymin:Cposy(i)+visionymax,...
        Cposx(i)-visionxmin:Cposx(i)+visionxmax);
    viewmatrixB=field.agentB.criminal(...
        Cposy(i)-visionymin:Cposy(i)+visionymax,...
        Cposx(i)-visionxmin:Cposx(i)+visionxmax);
    viewmatrix=viewmatrixA+viewmatrixB;

    maximum=max(max(viewmatrix));

    [badassy,badassx]=find(viewmatrix==maximum);

    % arrest one criminal if there is at least one present
    if maximum

        % if multiple agents have the same criminal level, chose 1 randomly
        random=randi(length(badassy));

        decision=zeros(size(viewmatrix));
        decision(badassy(random),badassx(random))=1;

        badassy=badassy-1+Cposy(i)-visionymin;
        badassx=badassx-1+Cposx(i)-visionxmin;

        prisoncountA=length(prison.agentA.position);
        prisoncountB=length(prison.agentB.position);
```

```
        if norm(decision.*viewmatrixA)
            move('A','field',badassx(random),badassy(random),...
                'prison',1,prisoncountA+1);
            remove(badassx(random),badassy(random),'A');

        else
            move('B','field',badassx(random),badassy(random),...
                'prison',1,prisoncountB+1);
            remove(badassx(random),badassy(random),'B');
        end
    end

end
end
```

Published with MATLAB® 7.10

```
function [] = release(jail_time_max)
%RELEASE Release prisoners.
%   Release prisoners.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables

global field prison sizex sizey

% number of prisoners

prisoncountA=length(prison.agentA.position);
prisoncountB=length(prison.agentB.position);

% find free positions

free=ones(sizey,sizex)-field.agentA.position-field.agentB.position-...
    field.cop.position;
[freeposy,freeposx]=find(free);

% count number of prisoners to release

releasecountA=0;

for i=1:prisoncountA
    if prison.agentA.jail_time(i)>=jail_time_max
        releasecountA=releasecountA+1;
    else
        break
    end
end

releasecountB=0;

for i=1:prisoncountB
    if prison.agentB.jail_time(i)>=jail_time_max
        releasecountB=releasecountB+1;
    else
        break
    end
end

% release the candidates

for i=1:releasecountA
    if isempty(freeposy)~=1

        % chose one free position randomly

        random=randi(length(freeposy));
        x=freeposx(random);
        y=freeposy(random);

        % remove free position from

        if length(freeposx)==1
            freeposx=[];
            freeposy=[];
        elseif random==1
```

```
            freeposx=freeposx(2:end);
            freeposy=freeposy(2:end);
        elseif random==length(freeposx)
            freeposx=freeposx(1:random-1);
            freeposy=freeposy(1:random-1);
        else
            freeposx=[freeposx(1:random-1); freeposx(random+1:end)];
            freeposy=[freeposy(1:random-1); freeposy(random+1:end)];
        end

        % move agent
        move('A','prison',1,1,'field',x,y);

        % remove prison cell

        prison.agentA.position=prison.agentA.position(2:end);
        prison.agentA.hardship=prison.agentA.hardship(2:end);
        prison.agentA.grievance=prison.agentA.grievance(2:end);
        prison.agentA.risk_aversion=prison.agentA.risk_aversion(2:end);
        prison.agentA.age=prison.agentA.age(2:end);
        prison.agentA.jail_time=prison.agentA.jail_time(2:end);

        % agents released from prison are neither criminal nor active

        field.agentA.criminal(y,x)=0;
        field.agentA.active(y,x)=0;

    end
end

for i=1:releasecountB
    if isempty(freeposy)~=1

        % chose one free position randomly

        random=randi(length(freeposy));
        x=freeposx(random);
        y=freeposy(random);

        % remove free position from

        if length(freeposx)==1
            freeposx=[];
            freeposy=[];
        elseif random==1
            freeposx=freeposx(2:end);
            freeposy=freeposy(2:end);
        elseif random==length(freeposx)
            freeposx=freeposx(1:random-1);
            freeposy=freeposy(1:random-1);
        else
            freeposx=[freeposx(1:random-1); freeposx(random+1:end)];
            freeposy=[freeposy(1:random-1); freeposy(random+1:end)];
        end

        % move agent

        move('B','prison',1,1,'field',x,y);

        % remove prison cell

        prison.agentB.position=prison.agentB.position(2:end);
        prison.agentB.hardship=prison.agentB.hardship(2:end);
        prison.agentB.grievance=prison.agentB.grievance(2:end);
        prison.agentB.risk_aversion=prison.agentB.risk_aversion(2:end);
```

```
        prison.agentB.age=prison.agentB.age(2:end);
        prison.agentB.jail_time=prison.agentB.jail_time(2:end);


        % agents released from prison are neither criminal nor active

        field.agentB.criminal(y,x)=0;
        field.agentB.active(y,x)=0;

    end
end
```

Published with MATLAB® 7.10

```
function []=movement(mobility,vision)
%MOVEMENT Lets agents move.
%   MOVEMENT(mobility,vision) prefixes a certain fraction of all agents
%   that is able to move, we call this fraction "mobility". The number of
%   movements is then received by a number of agents multiplied by
%   "mobility". Now the agents that are able to move are chosen. Every one
%   of them takes one step in one random direction within the agents
%   vision.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field sizex sizey

% only agents move
number_movements=round(sum(sum(field.agentA.position+...
    field.agentB.position))*mobility);

% locate agents
[Aposy,Aposx]=find(field.agentA.position);
[Bposy,Bposx]=find(field.agentB.position);
i=0;
while i<=number_movements

    if length(Aposy)+length(Bposy)>=1

        random=randi(length(Aposy)+length(Bposy));
        randomB=random-length(Aposy);

        if random<=length(Aposy)

            % agents A
            % find free spaces within viewfield
            visionxmin=vision;
            visionymin=vision;
            visionxmax=vision;
            visionymax=vision;
            if Aposy(random)-vision<=0
                visionymin=Aposy(random)-1;
            end
            if Aposy(random)+vision>=sizey+1
                visionymax=sizey-Aposy(random);
            end
            if Aposx(random)-vision<=0
                visionxmin=Aposx(random)-1;
            end
            if Aposx(random)+vision>=sizex+1
                visionxmax=sizex-Aposx(random);
            end

            [freeposy,freeposx]=find(...
                ones(visionymax+visionymin+1,visionxmax+visionxmin+1)...
                -field.agentA.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
                Aposx(random)-visionxmin:Aposx(random)+visionxmax)...
                -field.agentB.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
                Aposx(random)-visionxmin:Aposx(random)+visionxmax)...
                -field.cop.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
```

```
            Aposx(random)-visionxmin:Aposx(random)+visionxmax));


        freeposy=freeposy-1+Aposy(random)-visionymin;
        freeposx=freeposx-1+Aposx(random)-visionxmin;

        % move to one randomly chosen, free position
        if ~isempty(freeposy)
            randomfree=randi(length(freeposy));

            move('A','field',Aposx(random),Aposy(random),...
                'field',freeposx(randomfree),freeposy(randomfree));

            % remove agent from old position
            remove(Aposx(random),Aposy(random),'A');
            i=i+1;
        end
        % remove moved agent from list of agents, no agent is able to
        % move 2 steps
        if random==1
            Aposx=Aposx(2:end);
            Aposy=Aposy(2:end);
        elseif random==length(Aposx)
            Aposx=Aposx(1:random-1);
            Aposy=Aposy(1:random-1);
        else
            Aposx=[Aposx(1:random-1); Aposx(random+1:end)];
            Aposy=[Aposy(1:random-1); Aposy(random+1:end)];
        end

    else

        % agents B
        % find free spaces within viewfield
        visionxmin=vision;
        visionymin=vision;
        visionxmax=vision;
        visionymax=vision;
        if Bposy(randomB)-vision<=0
            visionymin=Bposy(randomB)-1;
        end
        if Bposy(randomB)+vision>=sizey+1
            visionymax=sizey-Bposy(randomB);
        end
        if Bposx(randomB)-vision<=0
            visionxmin=Bposx(randomB)-1;
        end
        if Bposx(randomB)+vision>=sizex+1
            visionxmax=sizex-Bposx(randomB);
        end

        [freeposy,freeposx]=find(...
            ones(visionymax+visionymin+1,visionxmax+visionxmin+1)...
            -field.agentA.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax)...
            -field.agentB.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax)...
            -field.cop.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax));

        freeposy=freeposy-1+Bposy(randomB)-visionymin;
        freeposx=freeposx-1+Bposx(randomB)-visionxmin;
```

```
            % move to one randomly chosen, free position
            if ~isempty(freeposy)
                randomfree=randi(length(freeposy));

                move('B','field',Bposx(randomB),Bposy(randomB),...
                    'field',freeposx(randomfree),freeposy(randomfree));

                % remove agent from old position
                remove(Bposx(randomB),Bposy(randomB),'B');
                i=i+1;
            end
            % remove moved agent from list of agents, no agent is able to
            % move 2 steps
            if randomB==1
                Bposx=Bposx(2:end);
                Bposy=Bposy(2:end);
            elseif randomB==length(Bposx)
                Bposx=Bposx(1:randomB-1);
                Bposy=Bposy(1:randomB-1);
            else
                Bposx=[Bposx(1:randomB-1); Bposx(randomB+1:end)];
                Bposy=[Bposy(1:randomB-1); Bposy(randomB+1:end)];
            end

        end
    else
        break
    end
end
```

Published with MATLAB® 7.10

```
function []=movement2(mobility,vision)
%MOVEMENT2 Lets agents move.
%   MOVEMENT2(mobility,vision)
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field sizex sizey

% only agents move
number_movements=round(sum(sum(field.agentA.position+...
    field.agentB.position))*mobility);

% initalize weighting matrix
[ii,jj]=meshgrid(1:2*vision+1,1:2*vision+1);
weighting=exp(-(ii-vision-1).^2/vision^2).*exp(-(jj-vision-1).^2/vision^2);
weighting(vision+1,vision+1)=0;

% locate agents
[Aposy,Aposx]=find(field.agentA.position);
[Bposy,Bposx]=find(field.agentB.position);
i=0;

while i<=number_movements

    if length(Aposy)+length(Bposy)>=1

        random=randi(length(Aposy)+length(Bposy));
        randomB=random-length(Aposy);

        if random<=length(Aposy)

            % find free spaces within viewfield
            visionxmin=vision;
            visionymin=vision;
            visionxmax=vision;
            visionymax=vision;
            if Aposy(random)-vision<=0
                visionymin=Aposy(random)-1;
            end
            if Aposy(random)+vision>=sizey+1
                visionymax=sizey-Aposy(random);
            end
            if Aposx(random)-vision<=0
                visionxmin=Aposx(random)-1;
            end
            if Aposx(random)+vision>=sizex+1
                visionxmax=sizex-Aposx(random);
            end

            [freeposy,freeposx]=find(...
                ones(visionymax+visionymin+1,visionxmax+visionxmin+1)...
                -field.agentA.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
                Aposx(random)-visionxmin:Aposx(random)+visionxmax)...
                -field.agentB.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
                Aposx(random)-visionxmin:Aposx(random)+visionxmax)...
                -field.cop.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
```

```
    Aposx(random)-visionxmin:Aposx(random)+visionxmax));


freeposy=[Aposy(random); freeposy-1+Aposy(random)-visionymin];
freeposx=[Aposx(random); freeposx-1+Aposx(random)-visionxmin];
decisionsum=zeros(length(freeposy),1);

for k=1:length(freeposy)
visionxmin=vision;
visionymin=vision;
visionxmax=vision;
visionymax=vision;
if freeposy(k)-vision<=0
    visionymin=freeposy(k)-1;
end
if freeposy(k)+vision>=sizey+1
    visionymax=sizey-freeposy(k);
end
if freeposx(k)-vision<=0
    visionxmin=freeposx(k)-1;
end
if freeposx(k)+vision>=sizex+1
    visionxmax=sizex-freeposx(k);
end

pers_weighting=weighting(vision-visionymin+1:vision+...
    visionymax+1,vision-visionxmin+1:vision+visionxmax+1);

decision=field.agentA.position(...
    freeposy(k)-visionymin:freeposy(k)+visionymax,...
    freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
    +field.agentB.position(...
    freeposy(k)-visionymin:freeposy(k)+visionymax,...
    freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
    *(field.agentA.grievance(Aposy(random),Aposx(random)))...
    -field.cop.position(...
    freeposy(k)-visionymin:freeposy(k)+visionymax,...
    freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
    *field.agentA.criminal(Aposy(random),Aposx(random));

decisionsum(k)=sum(sum(decision.*pers_weighting));

end

% move to one kly chosen, free position
[unused k]=max(decisionsum);
if isempty(k)
    k=1;
else
    k=k(randi(length(k)));
end

if k~=1
    move('A','field',Aposx(random),Aposy(random),...
        'field',freeposx(k),freeposy(k));

    % remove agent from old position
    remove(Aposx(random),Aposy(random),'A');
    i=i+1;
end

% remove moved agent from list of agents, no agent is able to
% move 2 steps
if random==1
    Aposx=Aposx(2:end);
```

```
            Aposy=Aposy(2:end);
        elseif random==length(Aposx)
            Aposx=Aposx(1:random-1);
            Aposy=Aposy(1:random-1);
        else
            Aposx=[Aposx(1:random-1); Aposx(random+1:end)];
            Aposy=[Aposy(1:random-1); Aposy(random+1:end)];
        end

    else

        % find free spaces within viewfield
        visionxmin=vision;
        visionymin=vision;
        visionxmax=vision;
        visionymax=vision;
        if Bposy(randomB)-vision<=0
            visionymin=Bposy(randomB)-1;
        end
        if Bposy(randomB)+vision>=sizey+1
            visionymax=sizey-Bposy(randomB);
        end
        if Bposx(randomB)-vision<=0
            visionxmin=Bposx(randomB)-1;
        end
        if Bposx(randomB)+vision>=sizex+1
            visionxmax=sizex-Bposx(randomB);
        end

        [freeposy,freeposx]=find(...
            ones(visionymax+visionymin+1,visionxmax+visionxmin+1)...
            -field.agentA.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax)...
            -field.agentB.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax)...
            -field.cop.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax));

        freeposy=freeposy-1+Bposy(randomB)-visionymin;
        freeposx=freeposx-1+Bposx(randomB)-visionxmin;
        decisionsum=zeros(length(freeposy),1);
        %%%%%
        for k=1:length(freeposy)
        visionxmin=vision;
        visionymin=vision;
        visionxmax=vision;
        visionymax=vision;
        if freeposy(k)-vision<=0
            visionymin=freeposy(k)-1;
        end
        if freeposy(k)+vision>=sizey+1
            visionymax=sizey-freeposy(k);
        end
        if freeposx(k)-vision<=0
            visionxmin=freeposx(k)-1;
        end
        if freeposx(k)+vision>=sizex+1
            visionxmax=sizex-freeposx(k);
        end

        pers_weighting=weighting(vision-visionymin+1:vision+...
            visionymax+1,vision-visionxmin+1:vision+visionxmax+1);
```

```
            decision=field.agentB.position(...
                freeposy(k)-visionymin:freeposy(k)+visionymax,...
                freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
                +field.agentA.position(...
                freeposy(k)-visionymin:freeposy(k)+visionymax,...
                freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
                *(field.agentB.grievance(Bposy(randomB),Bposx(randomB)))...
                -field.cop.position(...
                freeposy(k)-visionymin:freeposy(k)+visionymax,...
                freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
                *field.agentB.criminal(Bposy(randomB),Bposx(randomB));

            decisionsum(k)=sum(sum(decision.*pers_weighting));

            end

            % move to one kly chosen, free position
            [unused k]=max(decisionsum);
            if isempty(k)
                k=1;
            else
                k=k(randi(length(k)));
            end

            if k~=1
                move('B','field',Bposx(randomB),Bposy(randomB),...
                    'field',freeposx(k),freeposy(k));

                % remove agent from old position
                remove(Bposx(randomB),Bposy(randomB),'B');
                i=i+1;
            end

            % remove moved agent from list of agents, no agent is able to
            % move 2 steps
            if randomB==1
                Bposx=Bposx(2:end);
                Bposy=Bposy(2:end);
            elseif randomB==length(Bposx)
                Bposx=Bposx(1:randomB-1);
                Bposy=Bposy(1:randomB-1);
            else
                Bposx=[Bposx(1:randomB-1); Bposx(randomB+1:end)];
                Bposy=[Bposy(1:randomB-1); Bposy(randomB+1:end)];
            end

        end
    else
        break
    end
end
```

Published with MATLAB® 7.10

```
function []=movement3(mobility,vision)
%MOVEMENT3 Lets agents move.
%   MOVEMENT3(mobility,vision)
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables

global field sizex sizey

% only agents move

number_movements=round(sum(sum(field.agentA.position+...
    field.agentB.position))*mobility);

% initalize weighting matrix
[ii,jj]=meshgrid(1:2*vision+1,1:2*vision+1);
weighting=exp(-(ii-vision-1).^2/vision^2).*exp(-(jj-vision-1).^2/vision^2);
weighting(vision+1,vision+1)=0;

% locate agents
[Aposy,Aposx]=find(field.agentA.position);
[Bposy,Bposx]=find(field.agentB.position);

i=0;
while i<=number_movements

    if length(Aposy)+length(Bposy)>=1

        random=randi(length(Aposy)+length(Bposy));
        randomB=random-length(Aposy);

        if random<=length(Aposy)

            % find free spaces within viewfield
            visionxmin=vision;
            visionymin=vision;
            visionxmax=vision;
            visionymax=vision;
            if Aposy(random)-vision<=0
                visionymin=Aposy(random)-1;
            end
            if Aposy(random)+vision>=sizey+1
                visionymax=sizey-Aposy(random);
            end
            if Aposx(random)-vision<=0
                visionxmin=Aposx(random)-1;
            end
            if Aposx(random)+vision>=sizex+1
                visionxmax=sizex-Aposx(random);
            end

            [freeposy,freeposx]=find(...
                ones(visionymax+visionymin+1,visionxmax+visionxmin+1)...
                -field.agentA.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
                Aposx(random)-visionxmin:Aposx(random)+visionxmax)...
                -field.agentB.position(...
                Aposy(random)-visionymin:Aposy(random)+visionymax,...
                Aposx(random)-visionxmin:Aposx(random)+visionxmax)...
```

```
    -field.cop.position(...
    Aposy(random)-visionymin:Aposy(random)+visionymax,...
    Aposx(random)-visionxmin:Aposx(random)+visionxmax));


freeposy=[Aposy(random); freeposy-1+Aposy(random)-visionymin];
freeposx=[Aposx(random); freeposx-1+Aposx(random)-visionxmin];
decisionsum=zeros(length(freeposy),1);

for k=1:length(freeposy)
visionxmin=vision;
visionymin=vision;
visionxmax=vision;
visionymax=vision;
if freeposy(k)-vision<=0
    visionymin=freeposy(k)-1;
end
if freeposy(k)+vision>=sizey+1
    visionymax=sizey-freeposy(k);
end
if freeposx(k)-vision<=0
    visionxmin=freeposx(k)-1;
end
if freeposx(k)+vision>=sizex+1
    visionxmax=sizex-freeposx(k);
end

pers_weighting=weighting(vision-visionymin+1:vision+...
    visionymax+1,vision-visionxmin+1:vision+visionxmax+1);

decision=field.agentA.position(...
    freeposy(k)-visionymin:freeposy(k)+visionymax,...
    freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
    +field.agentB.position(...
    freeposy(k)-visionymin:freeposy(k)+visionymax,...
    freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
    *(field.agentA.grievance(Aposy(random),Aposx(random)))...
    -field.cop.position(...
    freeposy(k)-visionymin:freeposy(k)+visionymax,...
    freeposx(k)-visionxmin:freeposx(k)+visionxmax)...
    *field.agentA.criminal(Aposy(random),Aposx(random));

decisionsum(k)=sum(sum(decision.*pers_weighting));

end

% move to one kly chosen, free position
[unused k]=max(decisionsum);
k=k(randi(length(k)));

if k~=1
    move('A','field',Aposx(random),Aposy(random),...
        'field',freeposx(k),freeposy(k));

    % remove agent from old position
    remove(Aposx(random),Aposy(random),'A');
    i=i+1;
end

% remove moved agent from list of agents, no agent is able to
% move 2 steps
if random==1
    Aposx=Aposx(2:end);
    Aposy=Aposy(2:end);
elseif random==length(Aposx)
```

```
            Aposx=Aposx(1:random-1);
            Aposy=Aposy(1:random-1);
        else
            Aposx=[Aposx(1:random-1); Aposx(random+1:end)];
            Aposy=[Aposy(1:random-1); Aposy(random+1:end)];
        end

    else

        % find free spaces within viewfield
        visionxmin=vision;
        visionymin=vision;
        visionxmax=vision;
        visionymax=vision;
        if Bposy(randomB)-vision<=0
            visionymin=Bposy(randomB)-1;
        end
        if Bposy(randomB)+vision>=sizey+1
            visionymax=sizey-Bposy(randomB);
        end
        if Bposx(randomB)-vision<=0
            visionxmin=Bposx(randomB)-1;
        end
        if Bposx(randomB)+vision>=sizex+1
            visionxmax=sizex-Bposx(randomB);
        end

        [freeposy,freeposx]=find(...
            ones(visionymax+visionymin+1,visionxmax+visionxmin+1)...
            -field.agentA.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax)...
            -field.agentB.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax)...
            -field.cop.position(...
            Bposy(randomB)-visionymin:Bposy(randomB)+visionymax,...
            Bposx(randomB)-visionxmin:Bposx(randomB)+visionxmax));

        freeposy=freeposy-1+Bposy(randomB)-visionymin;
        freeposx=freeposx-1+Bposx(randomB)-visionxmin;

        % move to one randomly chosen, free position
        if ~isempty(freeposy)
            randomfree=randi(length(freeposy));

            move('B','field',Bposx(randomB),Bposy(randomB),...
                'field',freeposx(randomfree),freeposy(randomfree));

            % remove agent from old position
            remove(Bposx(randomB),Bposy(randomB),'B');
            i=i+1;
        end

        % remove moved agent from list of agents, no agent is able to
        % move 2 steps
        if randomB==1
            Bposx=Bposx(2:end);
            Bposy=Bposy(2:end);
        elseif randomB==length(Bposx)
            Bposx=Bposx(1:randomB-1);
            Bposy=Bposy(1:randomB-1);
        else
            Bposx=[Bposx(1:randomB-1); Bposx(randomB+1:end)];
            Bposy=[Bposy(1:randomB-1); Bposy(randomB+1:end)];
```

```
            end

        end
    else
        break
    end
end
```

Published with MATLAB® 7.10

```
function [] = cop_movement(mobility,vision)
%COP_MOVEMENT Moves Cops.
%   COP_MOVEMENT(mobility,vision) moves a number of randomly chosen cops
%   each to a new randomly determined position within his vision. The
%   number of movements is determined as in movement.m.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field sizex sizey

number_movements=round(sum(sum(field.cop.position))*mobility);

[posy,posx]=find(field.cop.position);

if isempty(posy)
    return
end

i=0;k=0;

[unused seq]=sort(rand(length(posy),1));

while i<=number_movements

    if k==length(posy)
        break
    end

    k=k+1;

    visionxmin=vision;
    visionymin=vision;
    visionxmax=vision;
    visionymax=vision;

    if posy(seq(k))-vision<=0
        visionymin=posy(seq(k))-1;
    end
    if posy(seq(k))+vision>=sizey+1
        visionymax=sizey-posy(seq(k));
    end
    if posx(seq(k))-vision<=0
        visionxmin=posx(seq(k))-1;
    end
    if posx(seq(k))+vision>=sizex+1
        visionxmax=sizex-posx(seq(k));
    end

    [freeposy,freeposx]=find(...
        ones(visionymax+visionymin+1,visionxmax+visionxmin+1)...
        -field.agentA.position(...
        posy(seq(k))-visionymin:posy(seq(k))+visionymax,...
        posx(seq(k))-visionxmin:posx(seq(k))+visionxmax)...
        -field.agentB.position(...
        posy(seq(k))-visionymin:posy(seq(k))+visionymax,...
        posx(seq(k))-visionxmin:posx(seq(k))+visionxmax)...
        -field.cop.position(...
        posy(seq(k))-visionymin:posy(seq(k))+visionymax,...
        posx(seq(k))-visionxmin:posx(seq(k))+visionxmax));
```

```
    freeposy=freeposy-1+posy(seq(k))-visionymin;
    freeposx=freeposx-1+posx(seq(k))-visionxmin;

    if ~isempty(freeposy)
        random=randi(length(freeposy));
        move('C','field',posx(seq(k)),posy(seq(k)),...
            'field',freeposx(random),freeposy(random));
        i=i+1;
    end

end

end
```

Published with MATLAB® 7.10

```
function [] = cop_control(cop_difference)
%COP_CONTROL Control the number of cops.
%   COP_CONTROL(cop_difference) increases or reduces the number of cops by
%   the value of cop_control by placing new cops at random free positions
%   on the field or removing randomly selected cops.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

global field sizex sizey numC IDC

for i=1:-cop_difference
    [Cposy,Cposx]=find(field.cop.position);
    if ~isempty(Cposy)
        random=randi(length(Cposy));
        remove(Cposx(random),Cposy(random),'C');
    end
end

for i=1:cop_difference
    free=ones(sizey,sizex)-field.agentA.position-...
        field.agentB.position-field.cop.position;
    [freeposy,freeposx]=find(free);
    if ~isempty(freeposy)
        random=randi(length(freeposy));
        field.cop.position(freeposy(random),freeposx(random))=1;
        IDC=IDC+1;
        field.cop.ID(freeposy(random),freeposx(random))=IDC;
        numC=numC+1;
    end
end

end
```

Published with MATLAB® 7.10

```
function [] = birth(birth_rate,age_max)
%BIRTH Simulate births.
%   BIRTH(birth_rate,age_max) chooses agents on the field that will be
%   duplicated to a new randomly chosen position as long as there is enough
%   free space. The agents are chosen by a birth probability which is the
%   ratio of birth_rate to age_max. By duplicating, all attributes are
%   copied except for age which is set to 0.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field sizex sizey IDA IDB IDC

birth_probability=birth_rate/age_max;
[freey freex]=find(ones(sizey,sizex)-field.agentA.position...
    -field.agentB.position-field.cop.position);
free=[freey freex];

%agent A
test=rand(sizey,sizex).*field.agentA.position;
[birthy birthx]=find(test>1-birth_probability);
birthA=[birthy birthx];

%agent B
test=rand(sizey,sizex).*field.agentB.position;
[birthy birthx]=find(test>1-birth_probability);
birthB=[birthy birthx];

births=[birthA; birthB];

[num cause]=min([size(births,1) size(free,1)]);

[~, seq_birth]=sort(rand(size(births,1),1));
[~, seq_free]=sort(rand(size(free,1),1));


for i=1:num
    if seq_birth(i)<=size(birthA,1)
        type='A';
    else
        type='B';
    end
    move(type,'field',births(seq_birth(i),2),births(seq_birth(i),1),...
        'field',free(seq_free(i),2),free(seq_free(i),1));
    eval(['field.agent',type,...
        '.age(free(seq_free(i),1),free(seq_free(i),2))=0;']);
    eval(['field.agent',type,...
        '.ID(free(seq_free(i),1),free(seq_free(i),2))=ID',type,';']);
    eval(['ID',type,'=ID',type,'+1;']);

end

if cause==2
    disp('not enough space for birth')
end

end
```

Published with MATLAB® 7.10

```
function []=death(age_max)
%DEATH Simulate natural deaths.
%   DEATH(age_max) removes all agents whose age exceeds age_max.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field

% locate oldest
[Aoldesty,Aoldestx]=find(field.agentA.age>=age_max);
[Boldesty,Boldestx]=find(field.agentB.age>=age_max);

% remove oldest from field
for i=1:length(Aoldesty)
    remove(Aoldestx(i),Aoldesty(i),'A');
end
for i=1:length(Boldesty)
    remove(Boldestx(i),Boldesty(i),'B');
end

end
```

Published with MATLAB® 7.10

```
function [] = move(type,from,x0,y0,to,x,y)
%MOVE Moves a single agent or cop.
%   move(type,from,x0,y0,to,x,y) duplicates an agent or cop determined by
%   type to a new position (y,x) on the field or prison which can be chosen
%   by to.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables
global field prison numA numB

% Agents
if strcmp(type,'A') || strcmp(type,'B')

    eval([to,'.agent',type,'.position(y,x)=',from,'.agent',type,...
        '.position(y0,x0);',...
        to,'.agent',type,'.hardship(y,x)=',from,'.agent',type,...
        '.hardship(y0,x0);',...
        to,'.agent',type,'.grievance(y,x)=',from,'.agent',type,...
        '.grievance(y0,x0);',...
        to,'.agent',type,'.ID(y,x)=',from,'.agent',type,...
        '.ID(y0,x0);',...
        to,'.agent',type,'.risk_aversion(y,x)=',from,'.agent',type,...
        '.risk_aversion(y0,x0);',...
        to,'.agent',type,'.age(y,x)=',from,'.agent',type,'.age(y0,x0);']);

    if strcmp(from,'field') && strcmp(to,'field')

        eval(['field.agent',type,'.active(y,x)=','field.agent',type,...
            '.active(y0,x0);',...
            'field.agent',type,'.criminal(y,x)=','field.agent',type,...
            '.criminal(y0,x0);']);

    end


    if strcmp(to,'field')

        eval(['num',type,'=num',type,'+1;']);

    end

    if strcmp(to,'prison')

         eval([to,'.agent',type,'.jail_time(y,x)=0;']);
    end

% Cops
elseif strcmp(type,'C')

    field.cop.position(y,x)=1;
    field.cop.position(y0,x0)=0;
    field.cop.ID(y,x)=field.cop.ID(y0,x0);
    field.cop.ID(y0,x0)=0;

end

end
```

Published with MATLAB® 7.10

```
function [] = remove(x,y,type)
%REMOVE Removes a single agent or cop.
%   REMOVE(type,x,y) deletes an agent or cop specified by type at the
%   position (y,x) on the field.
%
%   Date:
%       May 2010
%   Author:
%       Mundy Sam, Oberhauser Tim

% define global variables

global field numA numB numC

if strcmp(type,'A') || strcmp(type,'B')

    % clear agent on tile (x,y)

    eval(['field.agent',type,'.position(y,x)=0;',...
        'field.agent',type,'.active(y,x)=0;',...
        'field.agent',type,'.hardship(y,x)=0;',...
        'field.agent',type,'.grievance(y,x)=0;',...
        'field.agent',type,'.ID(y,x)=0;',...
        'field.agent',type,'.criminal(y,x)=0;',...
        'field.agent',type,'.risk_aversion(y,x)=0;',...
        'field.agent',type,'.age(y,x)=0;']);


    % reduce number of agents by 1

    eval(['num',type,'=num',type,'-1;']);

elseif strcmp(type,'C')

    % clear cop

    field.cop.position(y,x)=0;
    field.cop.ID(y,x)=0;

    numC=numC-1;

end

end
```

Published with MATLAB® 7.10