Migration Simulation on Potential Field based Landscapes

Jörg Dallmeyer
Goethe University Frankfurt
Robert-Mayer-Strasse 10
60325 Frankfurt, Germany
dallmeyer@cs.uni-frankfurt.de

Tjorben Bogon
Goethe University Frankfurt
Robert-Mayer-Strasse 10
60325 Frankfurt, Germany
tbogon@cs.uni-frankfurt.de

Yann Lorion
Goethe University Frankfurt
Robert-Mayer-Strasse 10
60325 Frankfurt, Germany
lorion@cs.uni-frankfurt.de

Ingo J. Timm
Goethe University Frankfurt
Robert-Mayer-Strasse 10
60325 Frankfurt, Germany
timm@cs.uni-frankfurt.de

ABSTRACT

In this paper, we present a novel migration simulation of entities over landscapes. The entities can explore the environment or "walk" from point to point, depending on their programmable attributes. For a fast adaptation to new maps, we have developed a converter which creates 3D-landscapes from 2D satellite (or map) images. Different altitudes get represented with different color codes and automatically converted into the landscape. The underlying system is based on a potential field, which is fully configurable and extendable for additional information about the landscape (e.g. weather, GIS). We show how this simulation can be applied to researches by simulating for example human migration over a specific landscape and how it helps to get information about the course of the walking-tour.

Categories and Subject Descriptors

I.6.3 [**Simulation and Modelling**]: Applications

General Terms

Migrants Simulation

Keywords

Migrants, Simulation

1. INTRODUCTION

The human migration in the early days of the humans out of Africa is only theoretically demonstrated. A simulation of the migration is complex because of the social behavior of humans. Interactions between the entities (human) and the geological environment influence every decision and make a real simulation nearly unfeasible. But archaeological excavations in places of discovery give an idea about the whereabouts of the humans and in which year they have populated which area. This provides an informative basis to track and approximate the path of the migration but does not explain the reasons for this migration. Different kinds of researches explain the biota, the weather and the animals living at specific decades in the early years. This provides a large base of data, which can help to understand the migration behavior, but until now no simulation concludes all this data at once. Many publications try to find a good model to describe how the humans came from Africa to Europe and develop simulators which are based on probability distributions or complex structured equations. In several simulators, the migration is based on population waves and not on the path they walk. We want to change the view on these simulators and to specify possible walking paths of migrations.

In this paper we show how we simulate the migration of entities over a landscape and compute a path between the places of discovery. We develop a model which is based on potential fields and provides the flexibility of being adapted to every environment settings or geographic information systems (GIS). The potential field allows concluding all datasets and combines them to a value which can be used as attraction of a place on a landscape. To simulate the migration in the pre age, our simulator computes a landscape out of a digital map. This map can either be a satellite map or a specific map where the biota is identifiable.

This paper is structured as follows. In chapter 2 actual simulations for human migration are discussed. After that we present our concept for a migration simulation in chapter 3. The functionalities and the GUI-methods of our simulation are presented in chapter 4. Chapter 5 shows a little example of using our simulation tool, before we finish with our conclusion and future workings in chapter 6.

2. MIGRATION SIMULATION

Different kinds of human migration simulations were developed in the past. Most simulations try to simulate broad-

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8642
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8642

ening of the humans in order to understand the migration behavior. Nikita and Nikitas understand human migration as a big random experiment [10]. Their simulation map consists of Africa and Europe and is reduced to 16,308 cells. Every cell has a probability to be populated by humans. This depends on the biota of the cell and whether humans populated the surrounding cells or not. Each cell consists of a large area of the landscape and can either be populated or not. There is no information about how many humans populated the cell. In each simulation iteration, each population can move, populate another cell or die with defined probabilities. Based on the randomness of the simulation, repeating experiments leads to different results. Different Experiments were performed where some cells were removed to see whether the human population path changed and where they walked through. The model which describes the earth does not change during the experiments, even though several thousands of years get simulated. The contents of the cells were the same at every time step. This is not authentic because the earth biota changed during the years. Continental drifts and for example volcanic activities changed the earth drastically.

A similar approach was introduced by Mithen and Reed [8] in 2002. The difference to Nikita and Nikitas was that the humans change their behavior depending on the cells on which they live. They could die, move to another cell or stay at the cell. The humans could adapt to the cell biota and live with the hot and dry weather or move to colder cells if they die in hot cells. The cells could change their content from land to water, this involved the evolution of the earth in the simulation. This approach is extended by [6, 5] with the genetic idea, that different population groups can exchange their attributes and the genetic migration can be monitored but no social structure is included.

Young et al. [16] developed an approach based on the Fisher equation for genetic diffusion.

$$\frac{\partial P}{\partial t} = RP - DP^2 + \nabla K \cdot \nabla P \qquad (1)$$

This equation combines time t, population density P, population growth rate R, population downturn D and the diffusion constant K. Young et al. assume that there is a maximum population density on every cell and the migration depends on the family. If humans travel to the next cell, they are never alone. Results of this migration simulation generated migration waves. The results of findings dated the human inhabitation of Australia in 50.000 BC and of Europe in 43.000 BC [11]. With this in mind Young et all. changed the R and P value and this helped understanding the out-of-Africa theory with their simulation. Although the simulation comprehends climatically changes and a good migration theory, there is no geo-information about the weather changes and the humans have no danger in the cells. They never died completely depending on aridity or dearth. Based on this equation a lot of other simulations were introduced. For example diffusion of nutrition production [1] or the diffusion of humans on north America [14]. In all these approaches, there is no possibility to connect time depending information like weather or other migration without changing the equation in a complex way. Another point is that only waves of migration are computed. The migration spreads in all directions and walks one or maybe two paths. There is no war or ethic barrier in the model.

The Open Geospatial Consortium[1] presents an open standard for geo-information systems to use geo-data for research. Data about the real environment like biota or weather data and maps from different GIS-systems can be used and shared with these developed standards. Paul Box [3] does not use an OCG-system, but he describes a way to combine a GIS with a multi-agent system to simulate the landscape changes whereby every entity (tree, cloud, etc.) is modeled as an agent. The advantages of multi agent systems are the attributes of the agents. An agent is flexible, proactive and social, which means that it can communicate with other agents. This social aspect is important for human migration, because every human acts depending on his behavior and his social environment. A project which is engaged with this social fact of the migration is done by Maerker [9].

Only the Agent-based simulations are continuously simulations. All other simulations are discrete and simulate the migration stepwise over cell-based landscapes. Every simulation considers the migration as population based. This means that if the humans "walk" over an area they populate it and stay there. In our view this is not right, because some areas are not habitable and humans hike to the next area without staying there forever. We want to change the view of the migration simulation to the path the humans walk and not to the populated areas. A big disadvantage of the discussed simulation is the inflexibility to integrate the social aspect and changeable landscapes. Every map has to be modeled in cells and the size of the cells is fixed. The humans contain no "brains" and are steered by the simulation. The results were enforced by adapting the parameters to steer the migration to specific points at specific time steps but there is no explanation why the migration is at these points. We want to develop a simulation where the entities are able to "think" and make their own decisions and the landscape is flexible and able to change in whatever it wants. This depends on the underlying potential field which converts data into a value which lets specific areas on the map be more attractive or abhorrent. Additionally, we want to simulate a migration on all kinds of maps and build a map converter which converts a landscape into a presentation for the simulation based on the color schema of the map and user settings.

3. CONCEPT OF OUR MIGRATION SIMULATION

This section describes the implementation of the simulation tool. At first, different land types of a map have to be classified. Based on the result of this step, a potential field gets computed on which simulation entities can be placed. At last, statistical export methods are introduced.

3.1 Classifying different cell-types in a satellite map

The implemented simulation tool imports a map, given by an image of an arbitrary format and calculates an internal cell representation of the imported map. A map-cell can e.g. be a range of pixels of an imported Bitmap. The only constraint is that the range must be a matrix-format, because

[1]http://www.opengeospatial.org/

the following methods work on matrices. Different image-types of a landscape are useable, because the classification of the cell-types is configurable. In our example, the functionality of the developed methods is shown using satellite maps. Maps for e.g. rainfall, heat-maps or maps for social welfare are thinkable, too.

The classification accuracy depends on the clarity of the partitioning between different characteristics in the given map. At first, the map has to be pre-processed with signal processing-operations, to find different characteristics in the map. We define a number of classes of landscape-types (e.g. "field", "forest", "mountain", "desert", "water" and "saltwater"). Each characteristic has to be placed in one of the defined classes.

Color layers and edge information of the given image are used to determine simple classification rules. Figure 1 shows the original image in part (a) and the different color layers red, green and blue in part (b) up to (d). Simple classification rules can be determined, regarding the colors of different areas. For example water has nearly no red intensity, but a high blue intensity and a green value near the middle of the intensity range (0-255). Water can be classified via edge information, too. To get a good classification accuracy, a combination of color channels, edge information and luminance of the map cells is used.

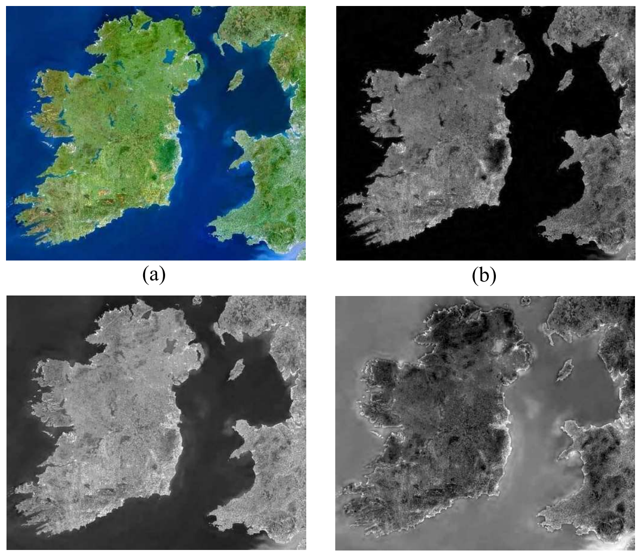
(a) (b) (c) (d)

Figure 1: Satellite image: Original, R-, G-, and B-channel

An edge is a color difference. To automatically find edges, we use the discrete convolution-operation adopted from [4] and shown in equation 2.

$$(g * h)(m,n) = \sum_{i \in h_x} \sum_{j \in h_y} g(i,j) \cdot h(i-m, j-n) \qquad (2)$$

The convolution is a standard operation in signal processing. It is used to manipulate a given signal g with another signal h with size h_x, h_y. Different filter-operations (e.g. blur filters, filters for edge detection or relief filters) can be implemented using the convolution. In our example, the Laplace-filter is used. Figure 2 (a) shows the result of a convolution of the map with a 5×5-laplacefilter (see [15]). All edges in the image now are emphasized, shown in this figure with light-colored pixels. A grayscale-image, calculated by averaging of the three color layers is used to get an idea about the brightness of the corresponding cells. This is shown in figure 2 (b).

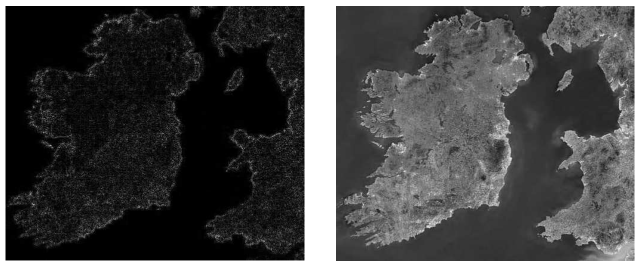
(a) (b)

Figure 2: Satellite image: Edges and luminosity

The calculated matrices are accessible for further calculations. As laplace$_{xy}$ we describe the result of the laplace-convolution operation on position $[x,y]$, shown in figure 2(a). The entries r_{xy}, g_{xy}, b_{xy} and gray_{xy} correspond to the entries of the RGB-layers and the grayscale-matrix of the given image.

Depending on these image operators, simple classification rules can be declared. In equation 3 and 4 the classification rules for the saltwater (sea) and forest class are shown. Other rules can be written by the user via adjusting the implemented ranges for R-, G- and B-channel, luminosity and edge intensity.

$$\begin{aligned} \mathrm{rule}_{\mathrm{sea}} &= \big(\mathrm{laplace}_{xy} < 5\big) \\ &\vee \big(\mathrm{gray}_{xy} < 51 \ \wedge\ \mathrm{r}_{xy} < 25\big) \\ &\vee \big(\mathrm{b}_{xy} > 100 \ \wedge\ \mathrm{g}_{xy} > 50 \ \wedge\ \mathrm{r}_{xy} < 30\big) \end{aligned} \qquad (3)$$

$$\begin{aligned} \mathrm{rule}_{\mathrm{forest}} &= \big(\mathrm{laplace}_{xy} > 5\big) \\ &\wedge \big(\mathrm{laplace}_{xy} < 20\big) \\ &\wedge \big(\mathrm{gray}_{xy} < 60\big) \end{aligned} \qquad (4)$$

In our example map, the forest cells appeared to have different green tones but certain edge intensity and a maximum luminosity. Equation 4 shows the result of this observance. These rules can be determined automatically by a simple point and click feature on the given map.

The classification is done by checking the classification rules for each cell. It is possible, that one cell can be classified by more than one rule. When the classification algorithm is called for one cell c, it determines the class of the cell by checking the rules in a given order and then recursively calls itself with the not yet classified neighbor cells of c and the determined class. When the classification algorithm is called with a cell and a class, it only checks this cell with the rule for the given class. When the rule is fulfilled, it classifies the cell with the given class, too and calls itself with the not yet classified neighbor cells of the given cell. By this way,

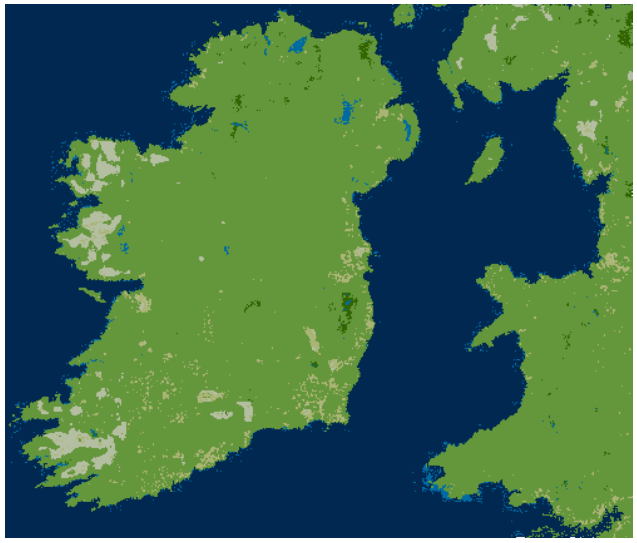

Figure 3: Classification result

Class	Potential
Saltwarter	255
Freshwater	200
Borderlines of freshwater areas	10
Desert	200
Field	75
Forest	100
Mountain	150

Table 1: LUT

connected areas can be classified. This recursively behavior of the algorithm is comparable to the well known *Grassfire-Algorithm* used for finding same support vector structures in an image or by using a variation of the region growing segmentation algorithm (see [15]).

Without appropriate prior knowledge, it is not possible to determine, whether a cell of water is saltwater or freshwater, because both can have identical characteristics. The classification algorithm is therefore called with each border cell of the map in the beginning, to check for saltwater. Normally, rivers and little seas are freshwater. In our example, these are attractive positions and classified as freshwater. Each cell, which cannot be classified by the developed rules, gets classified as "field". Figure 3 shows the result of the classification algorithm. Sea is shown dark-blue, lakes and rivers light-blue, field light-green, forest dark-green, desert brown and mountains gray.

Our example map uses the RGB color model. Maps which are given in another color model or display biota characteristics in another way, can be classified by adjusting the described methods.

3.2 Building a Potential field

The base for our potential field is a matrix, computed using a lookup table (LUT) (see [2]). Our LUT assigns a potential value to each class of landscape (see 1). A low potential value means for example, that the attraction of the corresponding cell is high, a high potential means vice versa. Shore areas of freshwater get a lower potential value as areas of freshwater, which do not belong to the border of this area, because for example hominids were able to drink water in these areas and rest there. The user is able to determine locations, which the hominids in the simulation have to visit. By this way, these places get a potential bonus of a constant value, e.g. -10. Predefined places obtain an attraction.

Transition regions between different cell types need to be attended. For example a desert will not fade to a field directly, but over an area. To obtain this, a 7×7 box filter (see [15]) is convoluted over the matrix. This results in a smooth potential field. Figure 4 shows the calculated potential field.

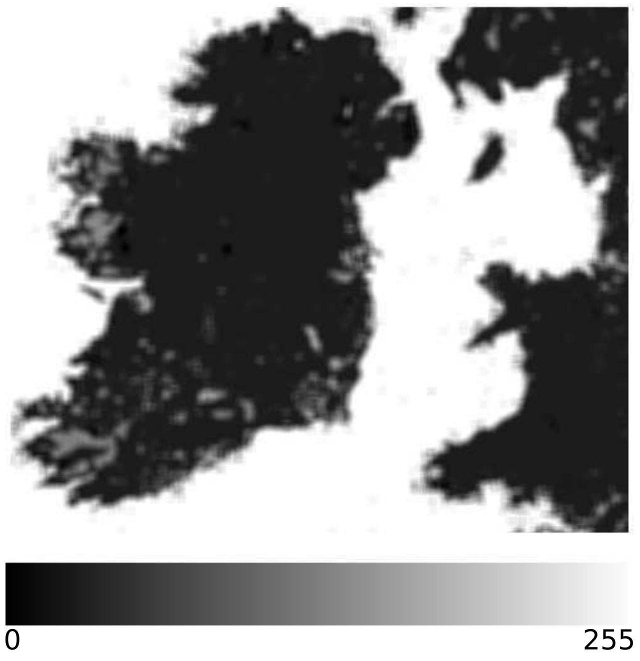

Figure 4: Potential field

The potential field is now represented by a $m \times n$-matrix P with the same dimensions (depending on the scaling), as the used satellite image. If a potential for coordinates $\vec{n} \in \mathbb{R}^2$ is needed, a bilinear interpolation from the four neighbor cells of $\vec{n}$ in P is calculated.

The potential field P can be manipulated during a simulation. If e.g. a volcanic eruption at cell $N = (x, y)$ is simulated, the area around N will have increased potential values after the eruption. A dry season can be simulated by increasing the potential values of all cells. By varying the potential field, many different influences on the simulated area can be modeled.

To steer entities over the potential field, a gradient-field is computed on the basis of the potential field. The gradient of a cell c is computed by examining a number of cells in the neighborhood of c and selecting the cell $\text{grad}(c) = c'$ where equation 5 is fulfilled.

$$P(c') \leq P(c'') \;\; \forall \; c'' \in \text{neighborhood}(c) \qquad (5)$$

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8642
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8642

The gradient vector is now calculated by subtracting the positions of c and c'. The neighborhood of c can be constructed in different ways. In the implemented simulation, a square with an edge length of 7 cells around cell c is used to compute the gradient of the cell.

If a cell is placed in an area with a constant potential value, no gradient can be computed. The gradient vector could have a length of zero then. In the implemented simulation, the vector then points to predefined positions in the potential field if a location is marked. The user can declare the positions. A possible appliance for this feature is the implementation of archaeological spots. By this way, some cells get more attracted, manually.

The simulation entities can have different behaviors concerning the predefined points. A fixed order of predefined points can be given and the entities can visit these points in the given order or randomly. The simulation ends, when the last spot has been visited by the entities or has become stopped manually by user.

These modes affect the determination of the position $D = \mathrm{grad}\,(c)$ for the calculation of the gradient of the cell c, if the gradient of c cannot be computed. The destination vector D can point to the next spot in a given order, to the spot with the lowest distance from the given cell or ignore all spots (have length zero).

Each simulation entity can find the best way from the cell it is located, by accessing the gradient vector of the corresponding cell. The gradient vector may influence the behavior of one entity, but this is not forced. The potential field can vary over the simulation time. By this way, different influences on the simulation entities can be simulated. If one entity manipulates the potential field, this will have an effect to all other entities, too. Via using this mechanism, an indirect communication between simulation entities can be accomplished.

3.3 Develope an Entity Behavior

This section describes how to get entities into the simulation. Each entity will be placed in the simulation via an interface. This interface defines a method, which is called by a control class of the simulation. The control class manages each part of the simulation. The simulation runs with discrete time steps. In each iteration, the control class calls the computation of each simulation entity behavior. Each simulation entity then computes its next state and returns its whereabouts back to the control class.

The implemented simulation uses as an example simple entities like simple reflex agents[12], which compute their position (external state) via combining the positions of their local best position, the global best position and a randomized vector over time-steps of the simulation. The global best position is identified via examination of the local best positions of all entities in the simulation. Additionally, the gradient vector of the current position of each entity influences their movements.

An interface for accessing the potential field and the gradient field is given to all simulation entities. When an entity, which represents a simple kind of hominid behavior in our example, is placed at a cell, it modifies the potential of the cell. For example, it exhausts resources. This can be modeled by incrementing the potential value of this cell. The cell regenerates its old potential over time. In each iteration, the potential values of all cells are updated to the direction of the original potential value of that cell before the simulation started.

3.4 Map-scaling and statistical export methods

To build a simulation for hominid movement, it is necessary to implement a map-scaling mechanism, which relates the movements of the simulation entities to real-world distances and real-world time. Thus the user needs to declare a map scale. In relation to the image-based classification, the user can state "x pixels correspond to y kilometers". With these given parameters, the covered distance of simulation entities can be estimated.

In addition to this specification, the user has the possibility to declare a maximum speed $v_{\max}$ for an entity. The control class scales movements down to $v_{\max}$, if the step of an entity is longer than $v_{\max}$. The maximum speed is declared by "each entity can move with a step size of $v_{\max}$ kilometers per iteration".

Different distance measures are calculated. At first, the direct connection lengths between the positions of the entities in the ascending iteration order. We use quadratic cells. A simple distance unit is the cell side length. It can also be exported in metric units, such as kilometers. It is possible that the movement path is not smooth and the entity jumps not directly from cell to cell. If this happens our simulation provides a *Bézier* interpolation method. It is used to make the movement path smoother. The calculated curve can be divided in transition pieces, whose length can be simply computed via Euclidean distance.

With the *Bézier*-method we have also the opportunity to compare the motion paths of different simulation cycles. We need this feature, because in our simulation model, the entity movements are calculated with a randomized factor. Each simulation run produces another movement path. The main problem at this point is, that different simulation cycles produce motion paths with different lengths. Motion paths cannot be simply scaled to a certain length. After [13, S. 330], the *Bézier* interpolation uses a number of $n+1$ sampling points $(P_i)_{i=0}^{n}$ to build a path, which starts in the first sampling point P_0 and ends in the last sampling point P_n. The remaining sampling points P_i attract the path to take its course near of them. The positions on the path can be computed by solving equation 6.

$$C\,(t) = \sum_{i=0}^{n} B_{i,n}\,(t)\,P_i \tag{6}$$

The function B is the Bernstein polynomial. The clue is, that the function C is only defined for the fixed domain of definition $t \in [0,1]$. The parameter t is used to define the position on the path, which shall be calculated. It is provable, that $C\,(0) = P_0$ and $C\,(1) = P_n$. All other t-values result in positions between P_0 and P_n. Thus the path can be scaled to a path with a fixed number $m+1$ of points shown in equation 7.

$$path = \left[C\,(0)\,, C\left(m^{-1}\right), C\left(2\cdot m^{-1}\right), \cdots, C\,(1)\right] \tag{7}$$

The described method allows to compare different motion paths. Aberrations of motion paths can now easily be calculated. An average motion path can be computed, by middling the motion paths of several simulation cycles.

Different statistics are provided for a running simulation. For example the average potential value for the simulation entities can be used as a measurement for the quality of the walked paths. If a group of entities shall be simulated, which has to find the way through the potential field collectively, the average distance to the middled positions of all entities gives a clue of how far the entities move from the center of the group.

A *MatLab* file is generated, which gives information about the simulation results. Boxplot-diagrams are plotted, which show the dispersion of the arrival times. The dispersion of the different orders of site visits is shown in another diagram (see figure 6). Vectors containing the arrival times at the last finding spot are given for further statistical analysis. By this way, different simulation setups can be easily checked against each other. *MatLab* can be used to conduct different statistical tests on the obtained data.

4. THE GRAFICAL USER INTERFACE (GUI) OF THE SIMULATION

This section gives an example of the implemented simulation tool. Figure 5 gives an idea about the appearance of the graphical user interface (GUI). In the top left-hand corner of the GUI, the control center can be found. The configuration of the GUI and the settings for the simulation can be done in this window. The buttons to start and stop the simulation and a time control for the simulation steps are offered as well. The speed of the simulation is influenced by the time length between each pair of simulation steps. Through the tabs of the control center the simulation ground settings like the shown migration paths in other windows can be configured.

On the bottom-side of the screenshot, all two-dimensional GUIs are allocated. In the bottom middle window, the used landscape map, in our example a satellite picture, is shown. If the user moves the mouse pointer over this window, an arrow to every pixel of the map demonstrates the gradient vector of the corresponding cell. The arrow points to the direction of the most attractive neighboring cell. If the simulation runs, a yellow line is painted and shows, depending on the settings, the real migration path or the *Bézier* interpolation. This is available on all 2D-windows, depending on the simulation setup.

Based on the configuration of the LUT, the classification map is shown on the left hand side of the bottom area of the GUI. This can be a control window to see that all cells got the right class. The classes differ by the chosen color. For example water is marked blue and the fields are tagged green. The right-bottom window demonstrates the computed potential field. With a grayscale-map, the different attractions of every cell, depending on the LUT get a value between white (very unattractive) and black (very attractive). This allows the user to check the changes on the potential field at every time step and makes the manipulation of the map easier. The last window is shown on the upper right.

The 3D-visualization of the landscape map combines the color of the basis landscape map and a height field. The height field gets computed with an additional LUT, which gives every determined class a fixed height. After this step, the height field gets convoluted with a blur filter, which results in the effect, that bigger mountains get a taller height. This visualization has the function to show the process of simulation in a realistic environment. As an alternative to the height field, the potential field can be used for 3D-visualization. The values of each cell can be frozen to the start potential value because the dynamic computation of the 3D visualization of the time variant potential field is based on texture mapping and this is very time complex. This feature can be switched on and off in the control center. It is possible to move the field of vision with mouse or keys over the 3D landscape. The user can cycle around or zoom in or out of the landscape. The quality of the values for the heightmap-LUT, results in the realism of the landscape (e.g. the heights of hills). It is easier to see the behavior of the entities if good values are chosen.

The finding spots are marked with a yellow point respectively box on all maps. The entities are red. On every step of the simulation, the movements of all entities are shown and depending on the user settings, the path of movement, too. Finally the user has a complete sight of all events of the simulation and can easily access the potential field with simple point and click on the maps.

5. EXAMPLE OF MIGRATION EVALUATION

To show how our simulation can be used for migration evaluation, we test different settings for our entity behavior and compare the results. As previously described, we cannot compare our results to other scenarios because all researches done in migration simulation simulate migration waves and not paths. The evaluation emphasis is on the quality of the motion paths of the simulation entities. We assembled different ratios for the evaluation.

At first, the average fitness values of each simulation entity are computed. As fitness, we describe the potential value of the actual cell of the entity. This ratio gives us an idea, how good the entities walk through the potential field. If the potential value is low, the cell is better for our entities. The ratio of one simulation itself gives no interesting information, but compared to the average fitness values for other settings in different simulation runs, an interesting result can be computed.

In our example implementation, we want to test if entities arrive at the sites earlier when they behave in a group or alone. We compare two groups of 40 entities with different behavior in 200 simulation runs. One entity group focuses on the global best value of the group and the other group focuses on the gradients of the cells. To let them walk predefined routes, we placed four markers on the landscape. If the simulation entities do not visit all predefined points in a given number of iterations, our example implementation fulfills a program abort. Concerning different setups for the entity behaviors of our simulation entities, the predefined points must be visited in a certain order or can be visited in any order. The resulting *MatLab* script shows the distribution of simulation runs concerning different orders of visit at the predefined places and the frequencies of occurrence of those. The distributions are plotted, automatically. An example is shown in Figure 6. The pros and cons of different

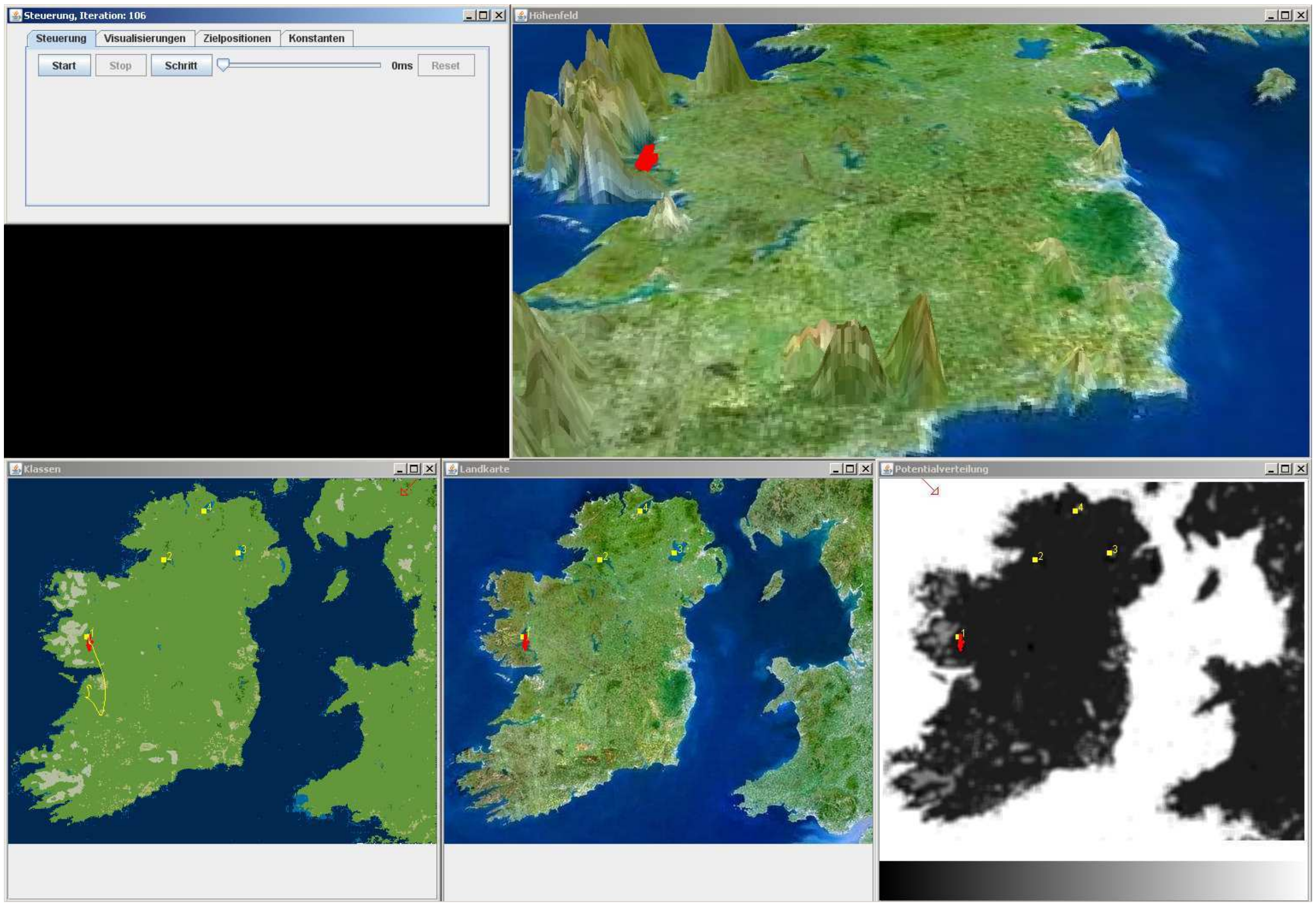

Figure 5: GUI

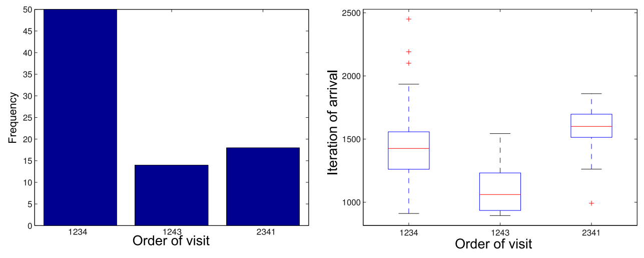

Figure 6: Diagrams of the frequencies and order of arrival

entity behaviors can be tested via examination of the *MatLab* script. In our example, we wanted the simulation entities to determine their way on the potential field as a group. To get information about the quality of this measure, the spreading rate of the entities, defined as the middled distances between the entity positions and the mean position of all entities are used.

Our simulation tool generates bitmap files, showing the motion paths of the simulation entities. The motion paths can be exported as the average motion path from multiple simulation runs, calculated via *Bézier* interpolating and middling of the related motion paths (see Figure 7).

The motion paths are split-up into sub paths from one spot to the next. Motion paths which did not visit all predefined positions are used, too. All motion paths which have the same order are used to compute a middled motion path. Figure 7 shows the middled motion path of the order 1,2,3,4. The predefined points are tagged with red quadrates. The motion path begins at the end of the path without a quadrate and ends in the contrarious end.

The calculated average motion path gives an idea of the way, the simulation entities walked. With this graphic, the user can decide if the path of the migration is right or if he wants to change the behavior of the entities.

With the defined map scale, all distances can be calculated in a metric unit, e.g. kilometers. Based on the distances, the propagation speed for the given map can be computed. In literature, different values for propagation speed are stated. The simulation entities of our model have a propagation speed of 0.11 km $\cdot$ year^{-1}, which is near the speed 0.12 km $\cdot$ year^{-1}, stated in [7]. The colonization period for the given map can be calculated. The result is given in the unit year.

We show that our model gives the opportunity, to simulate

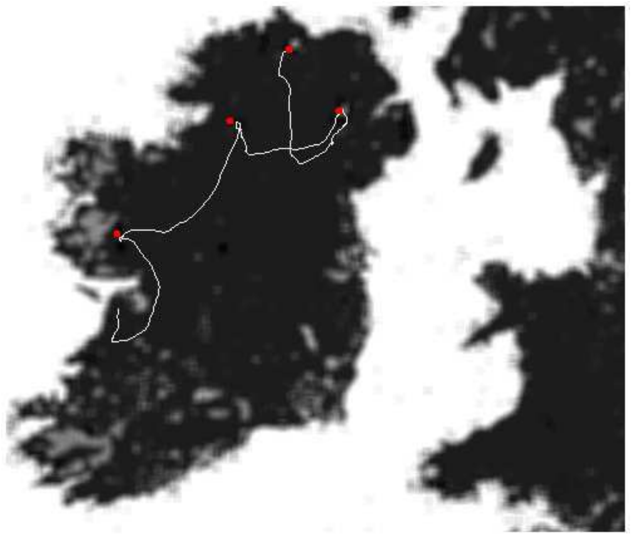

Figure 7: Motion path

different behaving entities and to compare the characteristics of the estimated motion paths.

6. CONCLUSION AND FUTURE WORK

In this paper we describe a novel migration simulation. Every kind of maps can be loaded and converted into a 3D landscape. User defined rules describe the landscape types and set the attraction for the classes. Supported by a potential field, every area of the landscape is flexible and changes its attraction for migration. This flexibility allows adapting every kind of data set to simulate events. The entities can communicate indirectly through the potential values. An easy behavior is possible depending on the gradient field which shows to the best field in the environment of each entity or to the next waypoint. The migration path is described by middling the positions of the human-group and results in waypoints over the landscape. The user sets predefined points in the landscape and assists the statistical analysis. The output of our simulation are data-structs for *MatLab* in order to get graphs about the arrival time and migration paths.

Our next steps are to build real agent structures for humans with different behavior and more social interaction between entities in the simulation. This behavior should base on social aspects and the team spirit of the early humans. With this extension the migration becomes more suitable. Another point is to involve the standards of the Open Geospatial Consortium in our calculation for the potential field. This data allows us to have a more realistic landscape and it is possible to test the migration on different biota and maps. We want to develop an interface for easy usage of our simulation and to give the opportunity to implement extensions of the behavior of the entities in a simple way. Last but not least, we want to improve the possibility to visualize the results. More interactions between the user and the landscape are possible. For example the user could create a storm or a dearth during the simulation to test migration with this obstacle.

7. REFERENCES

[1] G. J. Ackland, M. Signitzer, K. Stratford, and M. H. Cohen. Cultural hitchhiking on the wave of advance of beneficial technologies. *Proceedings of the National Academy of Sciences of the United States of America*, 104(21):8714–8719, Mai 2007.

[2] N. Bartelme. *Geoinformatik: Modelle, Strukturen, Funktionen*. Springer, Berlin, 4. edition, 2005.

[3] P. Box. Spatial units as agents: Making the landscape an equal player in agent-based simulations. In *Integrating Geographic Information Systems and Agent-Based Modeling Techniques for Simulating Social and Ecological Processes*, pages 59–82, Oxford, 2002. Oxford University Press.

[4] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, Thun, 6. edition, 2006.

[5] J. Hughes and S. Smith. Simulating global patterns of pleistocene hominin morphology. *Journal of Archaeological Science*, Februar 2008.

[6] J. K. Hughes, A. Haywood, S. J. Mithen, B. W. Sellwood, and P. J. Valdes. Investigating early hominin dispersal patterns: developing a framework for climate data integration. *Journal of Human Evolution*, 53(5):465 – 474, 2007. African Paleoclimate and Human Evolution.

[7] R. Köthe, J. Hennig, and F. Kliemt. *Was ist was Bd.9 Der Urmensch*. Tessloff Verlag Ragnar Tessloff GmbH & Co. KG, 2003. ISBN 978-3-7886-0249-9.

[8] S. Mithen and M. Reed. Stepping out: a computer simulation of hominid dispersal from africa. *Journal of Human Evolution*, 43:433–462, Oct 2002.

[9] M. Märker, V. Hochschild, and Z. Kanaeva. Multidisciplinary integrative georelational database for spatio-temporal analysis of expansion dynamics of early humans. *Computer Applications to Archaeology*, März 2009.

[10] P. Nikitas and E. Nikita. A study of hominin dispersalnext term out of africa using computer simulations. *Journal of Human Evolution*, 49(5):602–617, Juli 2005.

[11] R. G. Roberts, R. Jones, and M. A. Smith. Thermoluminescence dating of a 50,000-year-old human occupation site in northern australia. *Nature*, 345:153 – 156, Mai 1990.

[12] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

[13] P. Shirley, M. Ashikhmin, M. Gleicher, S. Marschner, E. Reinhard, K. Sung, W. Thompson, and P. Willemsen. *Fundamentals of Computer Graphics, Second Ed.* A. K. Peters, Ltd., Natick, MA, USA, 2005.

[14] J. Steele, J. Adams, and T. Sluckin. Modelling paleoindian dispersals. *World Archaeology*, 30(2):286–305, 1998.

[15] K. D. Tönnies. *Grundlagen der Bildverarbeitung*. Pearson Studium, München, 2005. ISBN 978-3-8273-7155-3.

[16] D. A. Young and R. L. Bettinger. Simulating the global human expansion in the latepleistocene. *Journal of Archaeological Science*, 22(1):89 – 92, 1995.