Fusing terrain and goals: agent control in urban environments

Varol Kaptan and Erol Gelenbe

Imperial College London, Exhibition Road, London SW7 2BT, UK;

ABSTRACT

The changing face of contemporary military conflicts has forced a major shift of focus in tactical planning and evaluation from the classical Cold War battlefield to an asymmetric guerrilla-type warfare in densely populated urban areas. The new arena of conflict presents unique operational difficulties due to factors like complex mobility restrictions and the necessity to preserve civilian lives and infrastructure. In this paper we present a novel method for autonomous agent control in an urban environment. Our approach is based on fusing terrain information and agent goals for the purpose of transforming the problem of navigation in a complex environment with many obstacles into the easier problem of navigation in a virtual obstacle-free space. The main advantage of our approach is its ability to act as an adapter layer for a number of efficient agent control techniques which normally show poor performance when applied to an environment with many complex obstacles. Because of the very low computational and space complexity at runtime, our method is also particularly well suited for simulation or control of a huge number of agents (military as well as civilian) in a complex urban environment where traditional path-planning may be too expensive or where a just-in-time decision with hard real-time constraints is required.

Keywords: Information Fusion, Navigation, Autonomous Agents, Urban Terrain

1. INTRODUCTION

The major change of context of modern military conflicts away from rural areas and into the confines of densely populated urban environments has posed unique challenges for tactical planning of military operations. These challenges require a new set of technologies and methods to enable efficient and effective operation within the constraints of urban environments as well as exploit as much as possible the specific features of the city terrain.

One of the new requirements is the ability to operate in an environment containing a large number of civilians where it may be very difficult to selectively target an enemy force. Failing to preserve civilian life can have dire consequences on the overall success of a military operation.

Another operational constraint is related to the requirement to preserve civilian infrastructure. Among other things, this constraint implies limitations on the mobility of forces. For example, the fact that the shortest path to a certain destination goes through residential or municipal property does not not automatically imply that a military vehicle can take that route at the expense of causing damage to civilian infrastructure.

The rapid advances in computing power in the last decades and the general availability of relatively cheap equipment makes computer simulation a promising tool for assessment of the applicability of new approaches designed for dealing with the constraints and limitations of the new battlefield as well as training of military personnel. The importance of simulation becomes even more obvious when the inherent high costs and the high risks of injury associated with real military exercises are taken into consideration.

In previous works[1, 2] we explored the possibility of modeling the group behavior of teams of autonomous agents with common goals through purely reactive methods like social potential fields.[3] One advantage of such methods is that they tend to scale very well when the number of mobile agents becomes very large. Another attractive feature of reactive agent control methods is the ability to build complex scenarios by defining a small number of broadly-defined goals. Instead of being explicitly defined, the behavior of individual agents in such systems become an emergent property. Unfortunately, such methods also have the disadvantage of being adversely affected by limitations of the terrain within which the agents are operating.

E-mail: v.kaptan@imperial.ac.uk, e.gelenbe@imperial.ac.uk

In this paper we introduce a method of agent navigation within a terrain containing non-convex obstacles which is particularly well suited for application in urban environments. Since our approach is based on the idea of transforming the real space into a virtual obstacle-free space, it can serve as an adapter layer for a number of control algorithms like the ones mentioned above, which while show promising results, are not well suited for navigation in an environment with many restrictions. It is interesting to note that while space deformations have been used extensively in many areas – for example animation and visual effects[4–6] or computer-aided modeling[7, 8] – we are not aware of any research into space deforming transformations as a tool for terrain navigation.

2. TERRAIN TRANSFORMATION

One of the main problems of predominantly-reactive behavior control techniques like the methods of simulating group behavior mentioned in the previous section is that these methods suffer from the local minima problem. This problem is common in mathematics when searching for parameters which globally maximize or minimize the value of a function - simple search approaches can easily get trapped at a local extremum point. In the problem of reactive agent navigation, getting stuck in a local minima can be due to either the particular agent configuration or extra constraints imposed on the agents by the terrain. The former is difficult to quantify because the force profiles involved in the agent model usually encode only the rough intent of a mission designer and maybe based on intuition. As such, they are quite subjective and it can be argued that a particular outcome was intended or can be considered good enough for a particular purpose.

In the rest of this paper we will focus on the second problem which is usually more troublesome, especially in urban environments. Let us illustrate the problem with a very simple example – suppose an agent is embedded in an environment where its motion is controlled through social potential fields or similar methods. The terrain includes some obstacles which have to be avoided. In the case when the collision avoidance scheme adversely affects the motion towards an intended goal, the agent may either become stationary or be stuck on a localized cyclic path which does not get it nearer to the final goal. When this was not the intended outcome, the agent has failed to achieve its goal.

2.1. Informal Description

Our approach to avoiding obstacles is based on the idea of finding a transformation between the real navigation space and a virtual obstacle-free space and applying the classical agent control methods within this new space. Before providing a mathematical description of our approach we we will try to illustrate our idea in a more informal but accessible as follows:

1. We assume that our terrain is represented as a 2D birds-eye view of the environment. All obstacles have a continuous boundary which is a closed contour. The interior of the obstacles is not accessible to agents. The space outside the contours is free space.
2. We start the process of transforming the real space into a virtual obstacle-free space by continuously shrinking the obstacle contours inwards until all the obstacles collapse into point singularities. The free space is *attached* to the contours and will *stretch* in the process of obstacle collapse.
3. When all the obstacles have collapsed, the new *stretched* free space would span the whole real terrain and any point in this *stretched* virtual space will be accessible from any other point trivially (by navigating along a straight line, for example).
4. It is very important that the transform conserves the local continuity of the space in the process – this property guarantees that any continuous paths (straight lines, for example) in the *stretched* space are continuous curves in the real space, and therefore, valid navigation paths.

2.2. Formal Description

It is quite easy to realize that there are infinitely many ways of performing the above transformation. What we will describe below is one approach which happens to be particularly easy to compute and is also well suited for obstacle avoidance.

Figure 1 shows an example of the types of terrains that we are interested in. The obstacles is represented as polygons with their interior painted in gray, their boundary outlined and a cross mark at what can be considered the obstacle center. This collection of shapes and their respective centers was randomly drawn by hand in order to provide a reasonable set of non-convex shapes that are commonly found in urban areas. Throughout the rest of this paper, we will use this data set in order to describe our approach and provide experimental results.

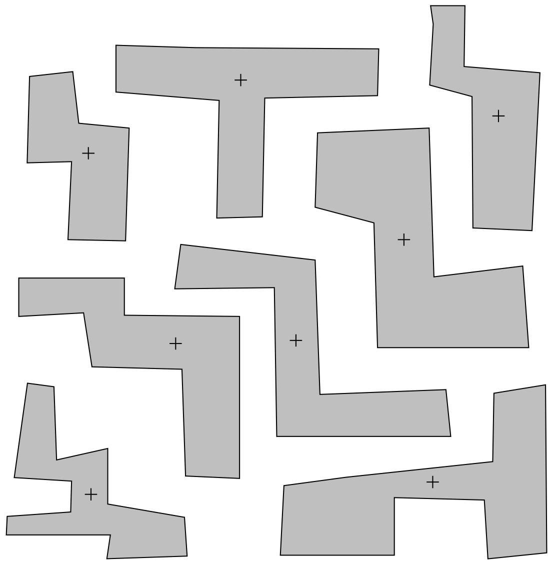


Figure 1. Terrain with a number of non-convex obstacles

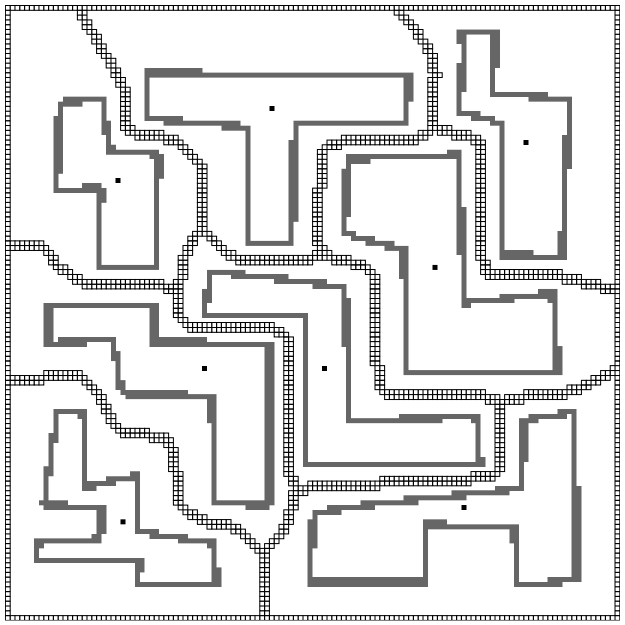


Figure 2. Boundary conditions (grid cells with fixed potentials)

2.3. Terrain Potential

The definition of the terrain transformation depends on a real-valued function over the terrain that we call the *terrain potential*. This potential can be computed in the following way:

1. We quantize the terrain into a grid. At the end of the computation each grid cell will have an associated potential (a real number between -1 and 1, inclusive).
2. Cells which contain obstacle centers have their potential fixed at -1.
3. Cells which include an obstacle boundary have their potential fixed at 0.
4. Cells that are completely inside or outside obstacles have their potential initialized to -0.5 and +0.5, respectively (this step is not necessary, but will help in faster convergence).
5. For each free cell (i.e. on the outside of obstacles), we compute the distance to the nearest obstacle. Cells which have neighbors closer to obstacles other than their own, have their potential fixed at +1. The same is done for cells at the boundary of the grid.
6. We run an iterative process where at each step the potential of each cell (excluding cells with already fixed potentials) is replaced by the average potential of its 4 neighbors. The iteration continues until the potential converges.

Figure 2 shows cells with fixed potential for our example terrain (quantized at 128×128). Cells with potentials fixed at -1, 0 and +1 are represented as black filled squares, gray squares and white outlined squares, respectively.

The formal mathematical explanation of the above process is that we are using Jacobi iteration to obtain a numerical solution of the Laplace equation subject to Dirichlet boundary conditions.

Physicists will easily recognize that the above procedure is equivalent to finding the electric potential ϕ of a system of electrical conductors (hence the name *terrain potential*), where cells at the center of obstacles correspond to a conductor with a potential fixed at $\phi = -1$, cells at obstacle boundaries correspond to a conductor with a potential fixed at $\phi = 0$, cells at equal distance between distinct obstacles represent conductors with $\phi = +1$ and all other cells represent free space (vacuum).

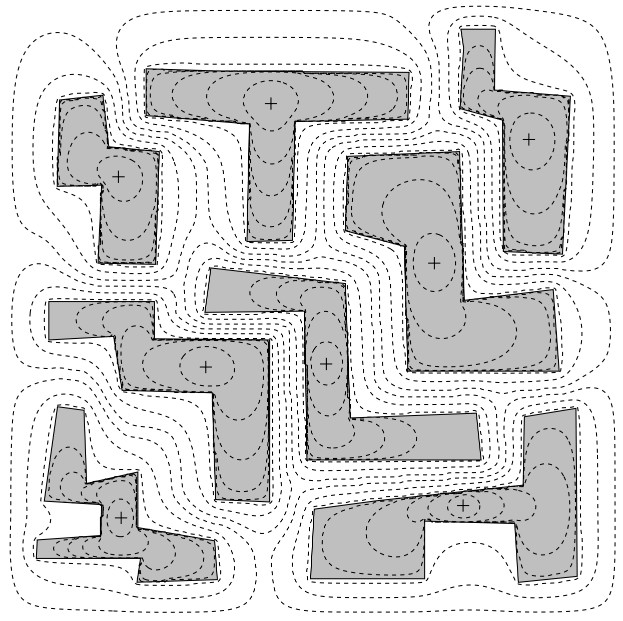


Figure 3. Equipotential profile

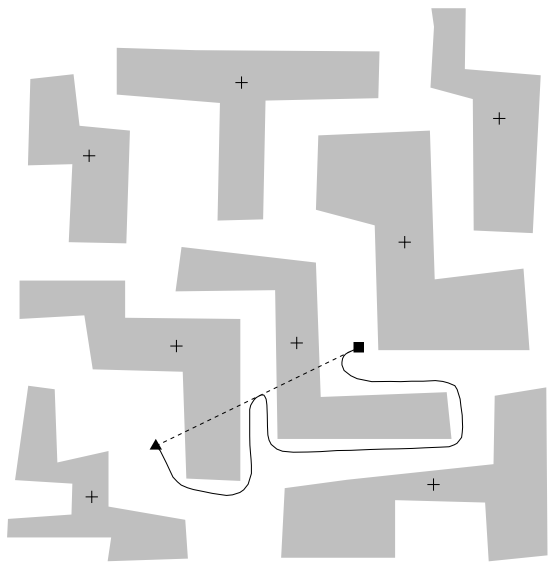


Figure 4. Shape of a virtual straight line in real space

Since the algorithms described later on operate in continuous space, we use quadratic filter[9] to reconstruct a continuous terrain potential from the quantized numeric solution.

Figure 3 shows the shape of the equipotential lines of ϕ for the test environment (quantization at 256×256, 2000 iterations). Note that the equipotential lines approximate the obstacle boundary at ϕ close to 0. As the potential decreases towards -1, the shape of the lines morphs smoothly towards a circle. Also, as the potential increases towards +1, the shape of the lines becomes more convex.

2.4. Transforming the terrain

Once the terrain potential has been obtained, we can find a transformation which will convert the real terrain into an obstacle-free virtual terrain. Note that the potential of the exterior of obstacles (i.e. accessible space) is limited to the range $(0; 1]$ and that of the interior of obstacles (inaccessible space) is in the range $[0; -1]$. The virtual position $\vec{p'}$ of any point $\vec{p}$ in the real terrain can be found* by moving down the gradient of the terrain potential (starting at $\vec{p}$) until a point $\vec{p'}$ with potential $\phi(\vec{p'}) = \max\{f(\phi(\vec{p})), -1\}$) is reached, where $f : \mathbf{R} \rightarrow \mathbf{R}$ is any function which is continuous and monotonically increasing in the range $[0; 1]$ and for which $f(0) = -1, f(1) = 1$ and $\forall x \in [-1; 0], f(x) \leq -1$. Since there are infinitely many functions with this property, we are actually defining a class of transformations isomorphic to the class of functions f with the above properties. For simplicity, from now on the term *transformation* will refer to any of one these.

Notice that when such a transformation is applied to all points in an obstacle interior it will collapse the obstacle to a point singularity coinciding with its respective center. Also, when applied to all other points, the transformation will expand free space to cover all of the terrain except a finite number of point singularities at the obstacle centers. Moreover, the transformation is reversible when the range is limited to free space.

The importance of these properties lies in the fact that instead of solving the problem of collision-free path between two points in the real terrain, we can instead solve the corresponding problem in the virtual space and apply the inverse transform to obtain a valid solution in the real space. The reason why this approach is attractive is that, due to the lack of

*Technically, this is true when there are no saddle points in the terrain potential. An example of when this can happen is an obstacle which is almost completely surrounded by another. This, however, is not a problem since we do not really compute the transformation itself.

obstacles in the virtual terrain[†], any continuous curve that connects two points in the virtual space corresponds to a valid path between their respective real counterparts. The simplest case of such a curve is a straight line. Figure 4 shows a sketch of how a straight line in virtual space might look when mapped back onto the real terrain. The starting point and the goal are represented with a triangle and a square, respectively. The dotted curve represents the shape of the straight line in real space.

At this point, we should note that while the computation of the terrain potential is a very straightforward and efficient procedure, obtaining a terrain transform from this potential is not. This is mainly due to the fact that gradient descent towards a point singularity over a uniform approximation of such potentials is numerically a very unstable.

Fortunately, computing such a transformation is not necessary. Nevertheless, we present it for the sake of providing a better understanding of why we use such an approach. For example, the path shown in Figure 4 can be generated by smoothly switching between (1) going towards a destination, (2) going around an obstacle (following the equidistant lines), and (3) back to going towards the destination after approaching the exit point at the other end of the obstacle. All the necessary information for this algorithm can be extracted from the local terrain potential. Note also that while not providing optimal paths, at least a collision-free path to destination is guaranteed (if one exists, of course). In the next section we will introduce a heuristic approach which can do better in terms of discovered path length at the expense of success ratio.

3. HEURISTIC NAVIGATION ALGORITHM

In this section we will describe a heuristic approach to agent navigation based on the idea of terrain potential. The algorithm described below is actually a quickly designed hack (mostly for the purpose of illustrating the idea and initial experimentation) and we are confident that a more careful and systematic construction will yield better results. Experimental results show that while not being able to guarantee a successful discovery of a path, this algorithm appears to performs quite well on our test terrain data.

The algorithm is based on performing a number of computations at each step. The outcome is a direction of motion and once the decision is applied the agent moves to the new position and the process is repeated until the destination is reached. These computations require the availability of the local terrain potential and the center of the closest obstacle. To be precise, the decision process depends on the following information:

- terrain potential at current position
- center of closest obstacle
- internal state variables (s and location of destination)

Note that the local nature of the required information allows it to be stored in completely distributed data structures. Table 1 describes our notation in detail.

Here is an algorithmic description of the steps necessary for obtaining a motion decision:

```
1. if a change in $\vec{c}$ is detected at the last step (*i.e. moved from one obstacle's field of influence into another*) then
   $s \leftarrow$ 'none'
2. if $\hat{n} \cdot \hat{d} \geq 0$ then *(agent is climbing the terrain potential – i.e. moving away from obstacle)*
   (a) if $\phi < 0.9$ then
       i. if $s =$ none then pick $s$ such that $\hat{d} \cdot \hat{s} > 0$
       ii. if $\hat{d} \cdot \hat{s} < -0.2$ then $\vec{a} \leftarrow \hat{s}$, otherwise $\vec{a} \leftarrow \hat{d}$
   (b) otherwise $\vec{a} \leftarrow \hat{d}$
3. otherwise *(agent is descending the terrain potential – i.e. moving towards the obstacle)*
   (a) if $s =$ none then:
```

[†]Except for point singularities which can be avoided trivially.

ϕ	terrain potential at the current location
$\hat{n}$	normalized gradient vector of ϕ (i.e. $\hat{n} = \frac{\vec{n}}{\|\vec{n}\|}$, where $\vec{n} = \vec{\nabla}\phi$
d	distance from current position to destination
d_{min}	threshold dependent on terrain scale
$\vec{d}$	a vector from current position to destination
$\hat{d}$	unit vector pointing towards destination
$\hat{c}$	unit vector pointing towards current obstacle center
s	current avoidance direction (left, right or none)
$\hat{s}$	a unit vector in the current avoidance direction (always perpendicular to $\hat{n}$)
$\vec{a}$	a vector representing decision outcome (i.e. direction of motion)

Table 1. Explanation of the notation used for describing the heuristic navigation algorithm

```
        i. if $\hat{c} \cdot \hat{d} < 0$ then pick $s$ such that $\hat{d} \cdot \hat{s} \geq 0$
       ii. else pick $s$ such that $\hat{c} \cdot \hat{s} \geq 0$
   (b) $\vec{a} \leftarrow \phi^2\hat{d} + (1-\phi^2)\hat{s}$
4. if $d < d_{min}$ then $\vec{a} \leftarrow b\hat{d} + (1-b)\vec{a}$, where $b = \sqrt{d/d_{min}}$
   (this operations modulates the behavior after the agent gets within $d_{min}$ of the obstacle)
5. at this point $\vec{a}$ contains the outcome of the decision process in the form of a direction of motion.
```

4. EXPERIMENTS

In this section we present experimental results on the performance of the heuristic navigation algorithm and provide quantitative comparison with other approaches. The terrain we used is shown in Figure 1 (terrain dimensions are 300×300 units, terrain potential computed in 2000 iterations at 256×256 quantization, $d_{min} = 10$ units).

Table 2 shows quantitative results based on a randomly selected set of 1000 missions. A *mission* is defined by a pair of points which represent initial and final positions of an agent. For each mission, we calculate the shortest path, the heuristic path and a path obtained from a simple reactive algorithm where an agent always moves in a direction which will minimize the distance to destination (or gets stuck if any action will actually increase the distance). We will refer to these algorithms as SHORTEST, HEURISTIC and SIMPLE. We chose to compare HEURISTIC with SHORTEST and SIMPLE because

- SHORTEST gives best results in terms of path length and success ratio but is computationally expensive and requires global terrain information,
- SIMPLE has very low computational complexity but very low success ratio too,

and we are looking for an algorithm which tries to combine the best properties of both ends of the spectrum (i.e. high success ratio and low computational complexity).

One of the shortcomings of HEURISTIC is that it has trouble approaching the final destination when it is very close to an obstacle[‡]. In order to analyze the effect of obstacle proximity we examine the subsets of missions for which the destination ϕ is bound by a lower threshold ϕ_{min}. Table 2 shows outcomes for ϕ_{min} ranging from 0 to 0.9. As it can be seen from these results, the success rate of HEURISTIC drops from 100% to 97.4% when ϕ at the destination is allowed to go below 0.2 and down to zero. Still, these results are much better than the success rate of SIMPLE which stays steady at about 35%, especially when we consider the fact that SIMPLE and HEURISTIC have the same computational complexity at runtime.

[‡]Step 4 in the algorithm description tries to minimize the effect of this problem, but does not remove it.

	Set 1	Set 2		Set 3					
ϕ_{min}	A_b	A_b	A_h	A_b	A_h	A_s	N_b	N_h	N_s
0	204.09	204.24	234.20	137.46	174.01	142.33	1000	974 (97.4%)	339 (33.9%)
0.05	205.21	204.98	233.75	141.90	154.50	146.93	928	921 (99.2%)	319 (34.3%)
0.1	204.92	204.96	232.33	142.63	155.38	147.64	883	879 (99.5%)	310 (35.1%)
0.15	204.03	204.07	230.89	143.33	155.13	148.36	850	846 (99.5%)	305 (35.8%)
0.2	205.44	205.44	232.23	144.86	156.84	150.00	814	814 (100%)	292 (35.8%)
0.3	206.49	206.49	231.63	145.08	157.70	150.10	727	727 (100%)	264 (36.3%)
0.4	207.22	207.22	232.42	147.74	159.90	152.97	657	657 (100%)	240 (36.5%)
0.5	209.30	209.30	234.08	149.75	162.44	155.25	559	559 (100%)	199 (35.5%)
0.6	209.74	209.74	234.57	153.93	165.91	159.64	487	487 (100%)	174 (35.7%)
0.7	209.68	209.68	233.86	151.69	162.13	157.09	404	404 (100%)	149 (36.8%)
0.8	212.47	212.47	237.87	155.45	167.42	161.01	310	310 (100%)	117 (37.7%)
0.9	210.65	210.65	233.64	162.06	172.90	167.38	196	196 (100%)	78 (39.7%)

Set 1	:	All missions for which $\phi \leq \phi_{min}$ at destination
Set 2	:	Subset of set 1 for which HEURISTIC was successful
Set 3	:	Subset of set 1 for which SIMPLE was successful
A_b	:	Average path length of shortest paths
A_h	:	Average path length for paths discovered by HEURISTIC
A_s	:	Average path length for paths discovered by SIMPLE
N_b	:	Number of missions for which $\phi \geq \phi_{min}$ at destination
N_h	:	Number of paths discovered (out of a total of N_b) by HEURISTIC
N_s	:	Number of paths discovered (out of a total of N_b) by SIMPLE

Table 2. Experimental results

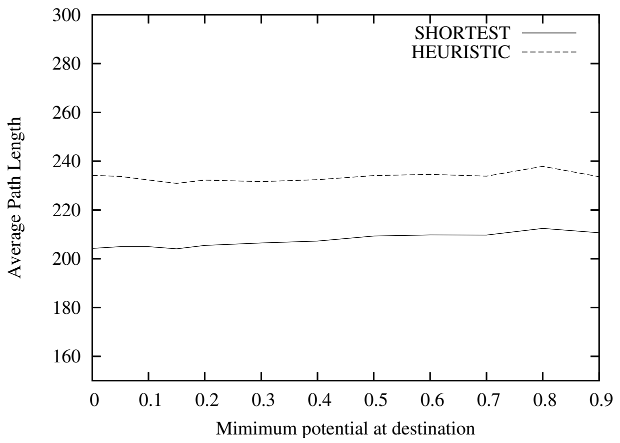


Figure 5. Average path length of set 2 missions w.r.t. ϕ_{min}

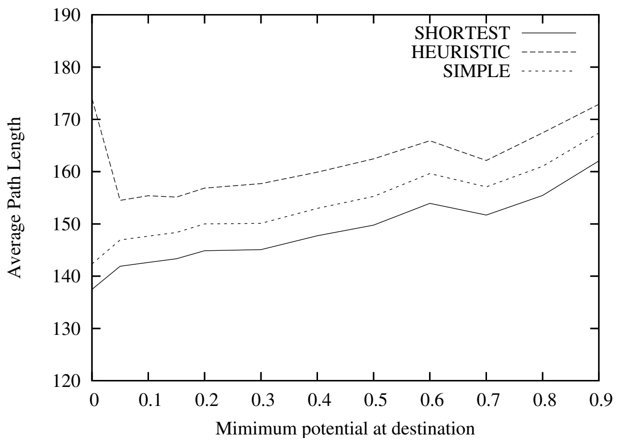
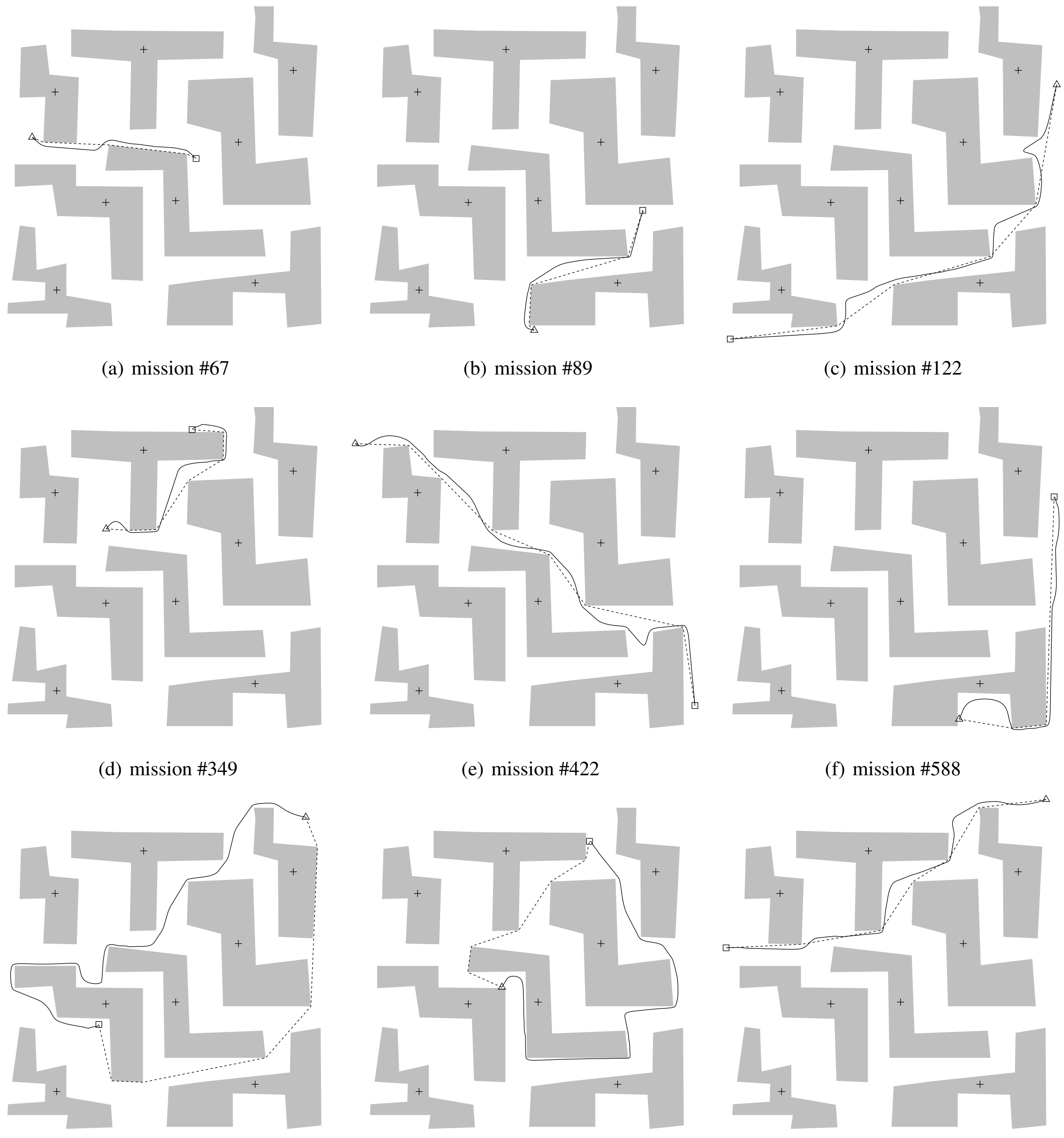


Figure 6. Average path length of set 3 missions w.r.t. ϕ_{min}

Figures 5 and 6 provide plots of the average path lengths from the Set 2 and Set 3 columns of table 2. According to figure 5, HEURISTIC is able to find paths which are on average 20% worse than the optimal. In the case of trivial paths (i.e. paths that can be discovered by SIMPLE), HEURISTIC finds paths which are on average about 10% percent worse than optimal while paths discovered with SIMPLE are on average about 5% worse than optimal. The problems HEURISTIC has when destinations are close to obstacle boundaries are also apparent in the far left part of figure 6. Finally, figure 7 shows nine randomly selected missions from the 1000 used in the experiments.



(a) mission #67 (b) mission #89 (c) mission #122

(d) mission #349 (e) mission #422 (f) mission #588

(g) mission #700 (h) mission #701 (i) mission #810

Figure 7. A random selection of mission outcomes. Initial and final positions are marked with a triangle and a square, respectively. Dotted lines represent SHORTEST paths and solid lines represent HEURISTIC paths.

5. CONCLUSION

In this paper we introduced a novel approach to path-finding and navigation in an environment with complex obstacles based on the idea of transforming a real terrain into an obstacle-free virtual space. We also presented as a proof of concept a heuristic algorithm which, by incorporating terrain characteristics into the decision process is able to significantly improve the success ratio of reactive agent navigation methods and at the same time retain their low run-time computational requirements. The advantage of the proposed way of abstracting relevant terrain information is that the amount of resultant data depends only on the dimensions of the terrain at a certain level of detail and can be computed and accessed in a completely distributed manner. Due to its low run-time requirements, our approach is particularly well suited for large-scale simulations and for navigation of autonomous agents with limited computational power. And last but not least, we believe that the problem of goal-based navigation is closely related to the inverse problem of discovering goals of agents based on observed motion, and we hope that our approach will contribute to a better understanding of agent mobility in complex environments and situation awareness.

REFERENCES

1. E. Gelenbe, K. Hussain, and V. Kaptan, "Simulating autonomous agents in augmented reality," *Journal of Systems and Software* **74**, pp. 255–268, Feb. 2005.
2. E. Gelenbe, V. Kaptan, and K. Hussain, "Simulating the navigation and control of autonomous agents," in *Proceedings of the Seventh International Conference on Information Fusion*, P. Svensson and J. Schubert, eds., **I**, pp. 183–189, International Society of Information Fusion, (Mountain View, CA), June 2004.
3. J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," in *International Workshop on Algorithmic Foundations of Robotics (WAFR)*, A. K. Peters, ed., pp. 431–459, (Wellesley, Massachusetts), 1995.
4. J. R. Kent, W. E. Carlson, and R. E. Parent, "Shape transformation for polyhedral objects," *Computer Graphics* **26**(2), p. 1992, 47-54.
5. H.-B. Yan, S.-M. Hu, and R. Martin, "Morphing based on strain field interpolation," *Computer Animation and Virtual Worlds* **15**(3-4), pp. 443–452, 2004.
6. K. Fujimura and M. Makarov, "Foldover-free image warping," *Graph. Models Image Process.* **60**(2), pp. 100–111, 1998.
7. D. D. models with local and global deformations: deformable superquadrics, "D. terzopoulos and d. metaxas," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(7), pp. 703–714, 1991.
8. J. Gain and P. Marais, "Warp sculpting," *IEEE Transactions on Visualization and Computer Graphics* **11**(2), pp. 217–227, 2005.
9. L. Barthe, B. Mora, N. Dodgson, and M. Sabin, "Triquadratic reconstruction for interactive modelling of potential fields," in *International Conference on Shape Modeling and Applications 2002 (SMI'02)*, p. 145, 2002.