Safe Pathfinding Using Abstract Hierarchical Graph and Influence Map

Conference Paper · November 2015

DOI: 10.1109/ICTAI.2015.125

4 authors, including:

DoHoon Lee
Pusan National University
49 PUBLICATIONS **297** CITATIONS
SEE PROFILE

2015 IEEE 27th International Conference on Tools with Artificial Intelligence

Safe Pathfinding using Abstract Hierarchical Graph and Influence Map

Daeseong Jong, Ickhwan Kwon, Donghyun Goo
Electronic and Computer Engineering
Pusan National University
Pusan, South Korea
{firefox, ickhwany, endeavor09}@pusan.ac.kr

DoHoon Lee
Electronic and Computer Engineering
Pusan National University
Pusan, South Korea
dohoon@pusan.ac.kr

Abstract— **Pathfinding is widely used in virtual environments, such as computer games. Most pathfinding types involve shortest pathfinding, which explores the fastest path, but tactical paths can also be searched for using various properties. This paper provides a method for finding safe paths that maintain a balance between path length and risks from hostile elements, as well as a method to reduce computation time using a hierarchical search strategy to enhance operational efficiency. Safe pathfinding uses the A* algorithm, referring to the influence map, which addresses the degree of risk in the terrain. The searched path represents its attributes regarding total length and cumulative risk.**

Keywords-pathfinding; influence map; abstraction

I. INTRODUCTION

Agents constantly move to achieve their objectives in virtual environments, such as computer games. Paths used by agents are determined by pathfinding algorithms. Generally, the most used pathfinding algorithm is the shortest pathfinding algorithm, because the shortest path takes the least amount of time and is economical. However, distance and time are not the only important properties. Suppose there are dangerous elements that have the potential to damage friendly agents. This may present a bigger economic loss than a delay if the agents are damaged. Therefore, safety is as important as distance and time in pathfinding. Safe paths normally avoid risky areas, so they can be long. Fast paths are the opposite, so they may involve taking risks to arrive at the destination. Although safe paths can be identical to fast paths, they are different in most cases. It is necessary to create paths that meet the criteria of safety and distance.

In this paper, we propose a safe pathfinding method for generating paths by taking into account safety and distance. We use the A* algorithm. The cost of A* in safe pathfinding is determined by evaluating the risk and path length in a virtual map. A* refers to the risk information in the Influence Map (IM). The IM is a technique that provides useful information about virtual environments. It divides a virtual map into several sections in a specific way, and each section provides numerical data representing the influence from the objects that are able to impact the agents. A* makes the paths by using an IM.

The search space of safe pathfinding is normally bigger than that of shortest pathfinding, since it considers two factors to create the paths. A method for maintaining the quality of the path as well as reducing the search space is needed. For this, we used the hierarchical search strategy. It performs abstraction, which makes a low-resolution graph from a high-resolution graph. A path is built from the abstract graph then refined through the high-resolution graph. This compensates for the defects of the safe pathfinding method.

II. RELATE WORK

The Influence Map [1] is a method of providing preferences for sections by analyzing the influence of objects in virtual environments. IM can have separate layers representing particular features. It enables agents to obtain the information they want by combining different layers. IM is used in many ways these days, because it is easy to use and intuitive. There are many ways to implement IM depending on the conditions. Although it is very simple, a Grid IM is an efficient and popular method. It is easy to implement and suitable for dealing with details. However, if the size of the terrain is big, it requires a lot of memory and considerable computation time, because the number of grids is proportional to the terrain size. Navigation meshes complement the weaknesses of the grid type. Normally, they represent traversable areas by dividing the virtual map into convex polygons. Although navigation meshes require less memory than the grid type, they perform well in pathfinding algorithms. Heckel et al. [2] proposed a way of representing the influence map using navigation meshes. A navigation mesh in IM inherits strengths from its use in pathfinding. If computational resources are limited, it would be better to use the navigation mesh. Lidén and Lars [3] used waypoints to evaluate the tactical value of a terrain. Like the navigation mesh, waypoints have good effectiveness in terms of memory, but weak detail.

IM can be utilized in a variety of situations. Wirth et al. [4] developed a Pac-Man AI based on IM. They measured the influence from game objects, such as enemies, items, and the locations of Pac-man agents, and decided where the Pac-Man agent would move by looking for the best positive locations in the terrain. The influence was adjusted by the weights on each IM variable. In order to improve performance, they tried to find optimal weights using the hill-climbing greedy algorithm. Straatman et al. [5] proposed various ways to apply IM, such as positioning, indirect fire, and suppressing fire, in computer games. Instead of saving data except for visibility into the database, it is computed on demand. Additionally, they use the waypoint graph to improve computational complexity. Miles et al. [6] represented the game state through

IM to evolve game players by using a genetic algorithm in a real-time strategy game. Jang and Cho [7] introduced a way to combine a neural network and a layered IM to evolve NPCs in a simulation game.

Graph search algorithms, such as A* [8] or Dijkstra's algorithm [9], are often used for tactical pathfinding. Danielsiek and Holger [10] utilized A* in IM to achieve natural and smooth agent group movements in a real-time strategy game. IM represents a location influence measured from ally and enemy, and A* finds the paths that are safe to agent groups using IM. Shi and Crawfits [11, 12] showed how agents travel to the destination safely by using obstacles and suppressing fire with A*. They recorded damage from enemies into IM and built a cover graph to illustrate the movement from the start point, around obstacles, and to the destination. A* uses the cover graph to find the optimal path.

Many studies have been conducted to increase the operational efficiency due to a growing requirement for immediate responses for finding paths. To enhance the algorithm performance, a hierarchical search has been proposed. This method first processes primary data, then refines the result using details. Shapiro et al. [13] utilized a two-level hierarchical network in a method using the hierarchical search strategy. This solution first finds the shortest paths from the start and end points to the nearest entry points to the next hierarchy level, and then searches for the fastest paths between these entry points. Sturtevant et al. [14] proposed a way to find the path based on spatial abstraction with partial path refinement. This abstracts the virtual map into a hierarchy. The abstract map in the hierarchy is then selected depending on resolution and details. The quality of the path is refined by using an abstract map from a low level.

III. Safe and Fast Path

A. Influence Map

Path length and safety have opposite properties. Agents need to take risks if they want to find fast paths. In contrast, agents that focus on safety must avoid dangerous areas. If paths that satisfy these two conditions exist, agents will naturally select them. Pathfinding algorithms have to find paths corresponding with agent preferences regarding safety and distance. Safe pathfinding requires a way to analyze a virtual environment to take many conditions into consideration. We use an influence map. IM is an extended version of the graph for pathfinding. The type of graph used for pathfinding usually has a grid structure, and each grid cell shows information about a location and links to neighbors. Apart from the navigation information, IM indicates values representing the influence measured from objects in virtual environments. The grid cell influence is the sum of the object's influence that affects a location corresponding to the grid cells. An analysis of IM is necessary to carry out a mission for agents. For example, suppose friendly agents try to run away from enemies. They will obtain the risk information of a position using IM, and then set a path around safe locations.

IM can be based on the navigation mesh or waypoint as well as the grid, because it inherits the pathfinding graph. The grid type could use considerable memory if it covers a large virtual map, but its merits include easy implementation and intuitiveness. We use the grid IM.

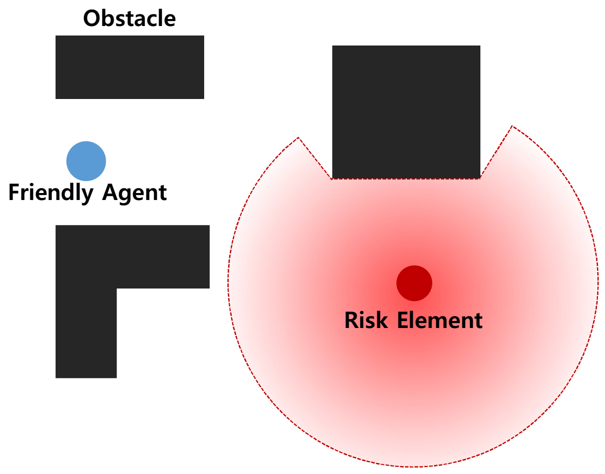

Figure 1. Main elements in virtual enviroment.

In this paper, a situation is set up that is as simple as possible to focus on a methodology for finding paths that are fast and safe from risks. Fig. 1 shows the primary elements. The friendly agent follows a path made by safe pathfinding. The path always starts from the friendly agent to the destination. The risk element is the threat to the friendly agent within a certain radius. The degree of risk and positions of all objects are stored in each corresponding IM grid cell. The factors affecting the risk assessment are as follows.

- Visual range of risk element.
- Danger radius of risk element.
- Degree of danger radius overlap.
- Obstacles.

Areas that cannot be seen near the obstacles within the danger radius of the risk elements are not affected by the risk. In Fig. 1, the danger radius of the risk element is represented as a red circle. The degree of risk is dependent on the distance from the center of the risk element.

$$g_r = e_d \cdot \frac{e_L - \text{distance}(g_p, e_p)}{e_L} \qquad (1)$$

In equation (1), g_r represents the degree of risk of a grid cell, and g_p is the position of a grid on the virtual map. e_d is defined as the maximum damage of the risk element. e_L is the danger radius of the risk element, and e_p shows the position of the risk element. Equation (1) expresses that the damage decreases linearly as the distance from the center of the risk element increases.

Algorithm 1 outlines the process of measuring the cumulative risk of a grid cell in the danger radius of the risk element. For every risk element defined in E, the algorithm checks that each grid cell is within the danger radius of a risk element, and if this is the case, the algorithm calculates the total damage defined in (1) of all the risk elements.

```
Algorithm1. Risk of Cell
Input.
cell            one cell of Influence Map
Output.
risk            expected damage
Constant, Variable, Function.
E               set of enemies
e               risk element
damage()        damage from risk element to cell

Procedure.
for all e ∈ E do
  if cell is covered by range of e then
    risk += damage(e, cell)
return risk
```

B. Influence Map

Safe pathfinding uses the A* algorithm, as does shortest pathfinding. The structure of IM is the graph, and the nodes of the graph correspond to the grid cells in IM. To obtain a safe path, we used the graph of IM.

$$\mathrm{f(i) = g(i) + h(i)} \tag{2}$$
$$\mathrm{m(i,j)} = w_s \cdot r(i,j) + w_d \cdot Dist(i,j) \tag{3}$$
$$\mathrm{h(i)} = w_d \cdot Dist(i,T) \tag{4}$$

A* determines the search order by evaluating the nodes of the graph using the heuristic cost function. Equation (2) defines the heuristic function in a basic form. g(i) is the total distance from the starting node to the current node, i. h(i) represents the expected distance from the current node, i, to the destination node. In equation (3), m(i, j) defines the distance between the connected nodes i, j. r(i, j) is the degree of risk, and Dist(i, j) is the Euclidean distance. h(i) uses Dist(i, T) to estimate the distance from node i to the destination node, T. h(i) does not consider the risk elements, so the heuristic function should not overestimate the path cost. In m(i, j) , w_s and w_d are weights that represent the importance of the safety and the distance. If w_s is high, the risk cost becomes expensive, so preference for the safe path arises even if the path length is long. On the contrary, a high w_d makes the agents prefer fast paths, because the distance cost is expensive even if they take a risk.

How the path choice varies can be seen in Fig. 2. As the travel cost between the neighboring cells in the terrain, the diagonal is 1.4, and the others are assumed to be 1. The value in the cell represents the degree of risk. The larger the w_s, the higher the probability of selecting path (a). On the other hand, the larger the w_d, the higher the probability of selecting path (b). Suppose that $w_s = 2, w_d = 1$. The cost of the safe path, (a), is 10.6, and the cost of the fast path, (b), is 17. In this case, path (a) will be selected. In general, the explored path in safe pathfinding goes through partially dangerous areas. For instance, if $w_s = 0.5, w_d = 1$, path (c) will be chosen, since the cost of (a) is 10.6, (b) is 11, and (c) is 10.3.

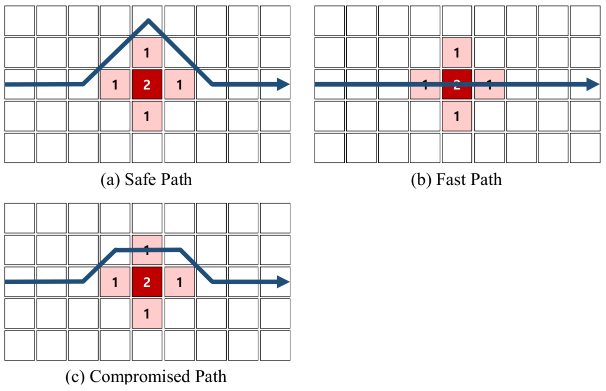

Figure 2. Path types with regard to importance of safe and distance

IV. HIERARCHICAL PATH FIDNING

A. Hierarchical Structure

Safe pathfinding usually explores broader areas than shortest pathfinding, since it considers two factors: Safety and distance. A broad search space requires high computational complexity. Therefore, it is necessary to enhance search effectiveness. A hierarchical search strategy is a good way of improving search efficiency. First, it performs a search using only the important features without considering the details, and it refines the details later.

In order to make the hierarchical structure, the base graph should be abstracted. Abstraction binds nodes that have similar properties to make one unit. This will be a node of the abstract graph. This process continues recursively until every node becomes one node in the most abstract graph. Fig. 3 shows a simple abstraction process. According to resolution, a certain number of nodes become one node in the next abstract graph.

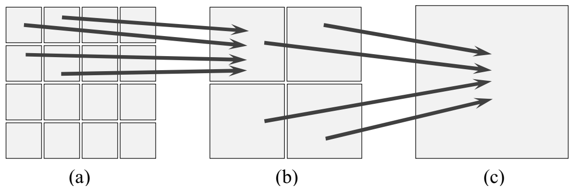

Figure 3. Abstraction concept. (a): Base graph. (b): Neighboring nodes are merged into a node of an abstract graph. (c): Maximum abstraction combines all nodes into one

It is not that every node will be abstracted. Nodes that only agents can move to can be abstracted. Nodes that agents cannot move to should be excluded from the abstraction. Therefore, we use the clique-based abstraction method suggested by Sturtevant et al. Fig. 4 represents the process, in which black nodes represent inapproachable positions. Clique refers to the child nodes of a node in an upper-level abstract graph. A clique should be a completed graph and consists of up to four nodes. In (a), c1 represents a 4-clique composed of four nodes. After the 4-clique becomes a node of the upper abstract graph, the algorithm searches for a 3-clique. The 4-clique combines with a node in the upper abstract graph and the 3-cliques c1, c2, and c2 are shown in (b). The 3-clique becomes a node in the upper abstract graph, like the 4-clique. 2-cliques undergo the same process. Nodes that do not belong to any clique are called orphans. If only one neighboring clique of an orphan exists, the orphan will be combined with the neighboring clique. An orphan that has more than two neighboring cliques becomes a node of the upper abstract graph only. In (c), the orphans o1 and o2 have only one neighboring clique, so they combine with the neighboring clique. (d) is a completed abstraction state. A node of an abstract graph represents their child nodes in a low-level graph, so its coordinates are the mean of the child's coordinates. Additionally, its risk is the mean of the child's risk.

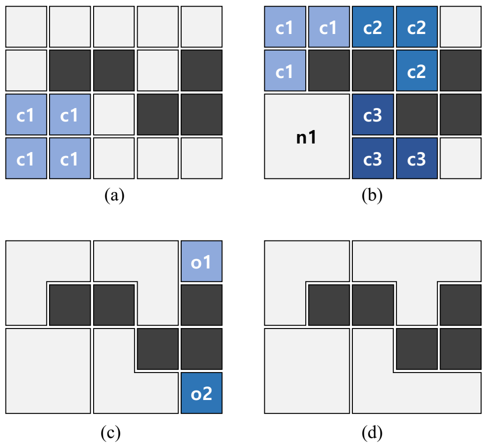

Figure 4. Clique based abstraction

B. *Safe Pathfinding*

Safe pathfinding uses a hierarchical abstract graph to improve computing speed. It starts by choosing the abstract graph level and then finds the nodes that belong to the start and destination nodes in the base graph. Safe pathfinding progresses from the chosen abstract graph level to the base graph recursively. The path found in the upper-level graph restricts the search space of the low-level graph to its trace. In other words, safe pathfinding performs in the child nodes of the upper-level path's nodes.

In Fig. 5, (a) represents pathfinding in the upper level and (b) shows the restricted search space of the low-level graph. It is assumed that all nodes are connected in four directions with neighboring nodes. In (a), every node can be explored except for the black nodes. The orange nodes were explored through the pathfinding, and the blue nodes are a path from the start point to the destination. In (b), nodes that can be explored appear as white nodes. It can be seen that only the child nodes of the upper-level path can be explored.

The hierarchical level of the abstract graph determines the detail of the path and computational complexity. If the level of the abstract graph is high, the computational complexity is low, but the detail of the path can be poor. Conversely, a low-level abstract graph can have fine detail, but have adverse effects on computational complexity. It should take into account the complexity of the terrain in order to minimize the loss of operation speed and accuracy. Abstraction is performed to lower the resolution of the graph. Complex terrain expressed in a low-resolution graph causes information loss. In this case, it may as well choose the low-level abstract graph to suppress detail loss. Less complex terrain can reduce computational complexity by using a higher level abstract graph. However, this is not always satisfied. The pathfinding in the abstract hierarchical graph is performed from a higher level to a lower level in a hierarchical fashion. Therefore, pathfinding that starts in the higher level may explore more nodes than in the lower level. If this is the case, the accuracy and the operation speed both suffer. It is important to find a proper abstract level.

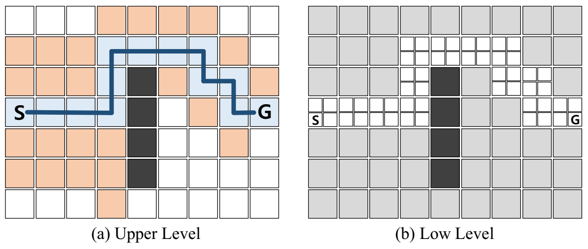

Figure 5. Path finding in abstract hierarchical graphs. (a): Pathfinding in upper level abstract graph (b): Pathfinding in low level abstract graph is conducted in the trace of the path of (a)

V. EXPERIMENTS

Fig. 6 represents the starting point, destination, friendly agent, enemies, and obstacles in the virtual map. Enemies that are risk elements have a 4-m danger radius from their centers. The degree of risk increases linearly from the edge to the center of the danger radius, and the risk value of the center is set to 33. The size of the virtual map is $400m^2$. The IM uses 10,000 grids. The program was made with a unity engine. All experiments were run on an Intel Core i3-4150 3.5GHz with 8GB of Memory.

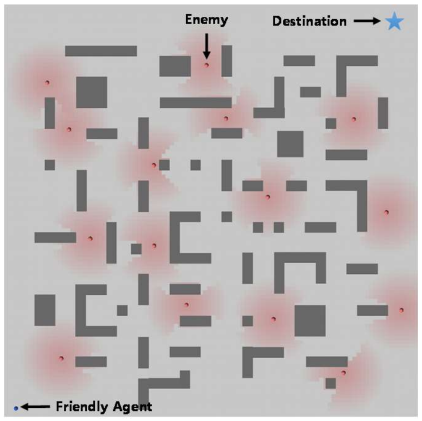

Figure 6. Friendly agent, enemies and destination on the virtual map. Safe pathfinding is performed through refering to the risk from enemies and distance to the destination.

A. *Fast path, Safe path*

We conducted the experiments to determine how the path changes depending on safety, distance, weight, and abstract level. Fig. 7 shows the distance and cumulative risk in respect to the safety weight. As safety and distance have opposite characteristics, we only changed the safety weight while fixing the distance weight at 1. Distance and cumulative risk are the average of the measured values at hierarchical levels of 0–3. When the safety weight increases, the distance increases and the risk falls. The risk decreases drastically between the safety weights 0 and 1, and the reduction ratio declines as the safety weight increases. The distance increases when the safety weight increases, but the variation rate is lower than the risk's. This implies that the advantages of risk reduction are significant compared to the disadvantages of the distance increment when the safety weight increases.

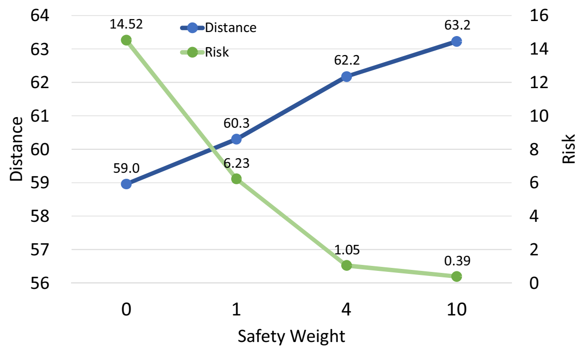

Figure 7. Distance and degree of risk over safety weight

The differences in risk and distance in accordance with hierarchical level when the weights of distance and safety are given are shown in Fig. 9. The hierarchical level 0 gives the best details, since it is a base graph. (a) is the shortest pathfinding, because the safety weight is 0. In this case, the difference in distance between the 0 and 3 hierarchical levels is 3.36. This is bigger than other hierarchical levels. The variation in risk in all the experiments is significant compared to the distance. The largest variation in risk shown in (b) represents 14.02 as an average variation at hierarchical levels of 0–3.

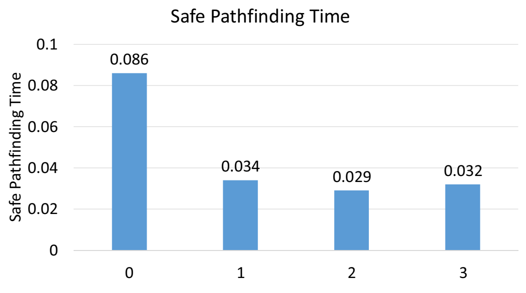

Figure 8. Distance and degree of risk over hierarchical level. The higher hierarchical level, the greater abstraction.

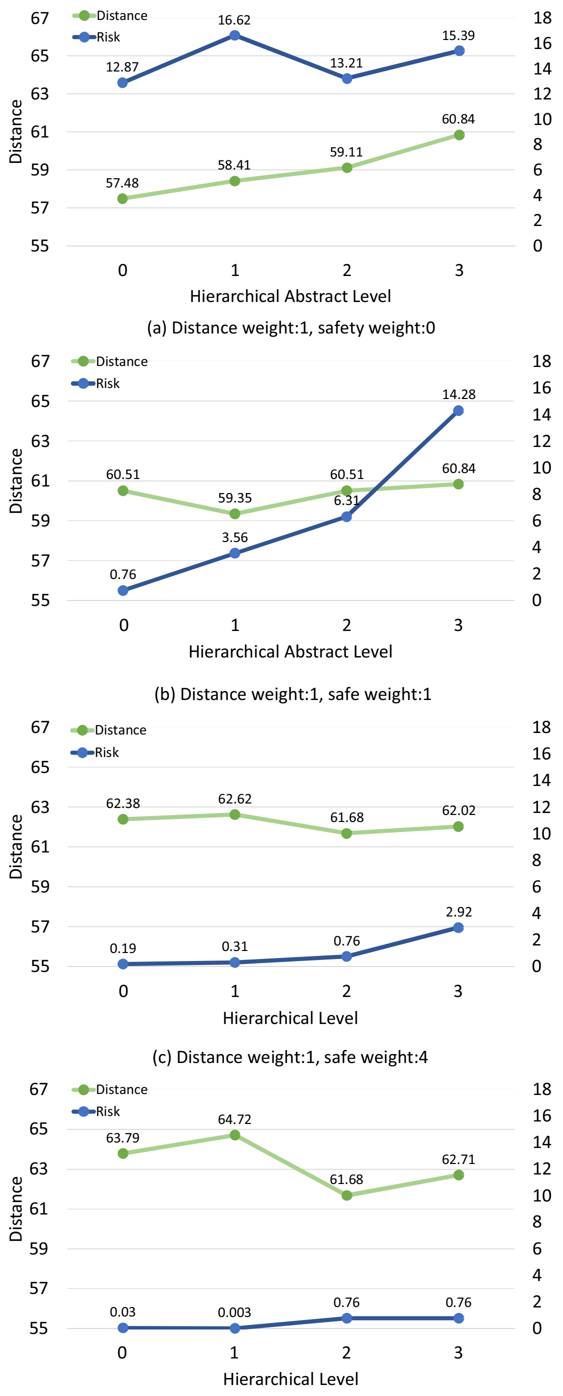

Figure 9. Amount of time safe pathfinding took in different abstarct hierarchical level.

B. *Abstraction time*

The waypoints that the agent will pass by are shown in Fig. 10. Safe pathfinding explores the path to the final location through each waypoint.

Fig. 8 represents the exploration time at each hierarchical level when the distance and safety weights are set to 1. Hierarchical level 0 shows the biggest operation time, because it is the base graph. Hierarchical level 1 takes a smaller operation time than hierarchical level 0. This means the abstraction reduces the exploration terrain. Hierarchical level 2, which was compared to hierarchical level 1, reduces the operation time, but the amount of reduced time is smaller than the variation between level 0 and level 1. Hierarchical level 3 takes more time than level 2. This implies that safe pathfinding from level 3 explores more nodes than at level 2. If a specific hierarchical level has less details as well as broader search areas than a lower level, it is better to exclude it.

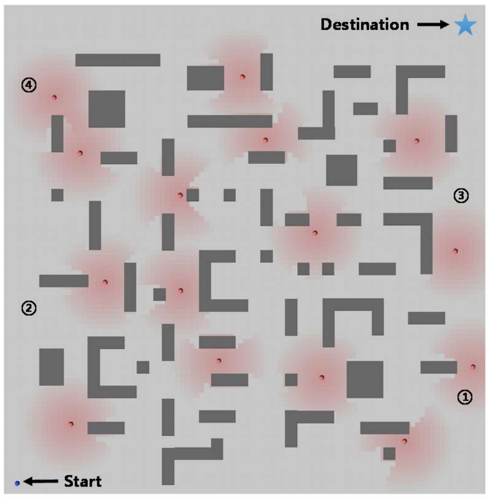

Figure 10. Path leading to destination through waypoints ①–④.

VI. Conclusion

In this paper, we proposed a safe pathfinding method to find a safe as well as fast path in virtual environments. Influence maps are used to measure the degree of risk associated with risk elements on the virtual map. Safe pathfinding explores the path considering the influence of each grid and the distance from the friendly agent to the destination. In IMs, the influence of a grid can change depending on the weights of the safety and the distance to help find paths that balance safety and distance.

Mostly, safe pathfinding should explore broader terrain than shortest pathfinding. In order to reduce the computational complexity, we used a hierarchical search strategy. This can lower details, but has the advantage of reducing the terrain to be searched. Our study indicated that it caused detail loss, but improved computational complexity.

As a future work, we want to find the optimal weights of distance and safety for a given condition. This could suggest an optimized path by default. To achieve this, we will introduce additional parameters and use machine learning.

References

[1] P. Tozour, “Influence Mapping,” Game Programming Gems 2, 2001.

[2] Heckel, Frederick WP, G. Michael Youngblood, and D. Hunter Hale. "Influence points for tactical information in navigation meshes." Proceedings of the 4th International Conference on Foundations of Digital Games. ACM, 2009.

[3] Lidén, Lars. "Strategic and tactical reasoning with waypoints." AI Game Programming Wisdom, Charles River Media. 2002.

[4] Wirth, Nathan, and Marcus Gallagher. "An influence map model for playing Ms. Pac-Man." Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On. IEEE, 2008.

[5] Straatman, Remco, and Arjen Beij. "Killzone’s AI: dynamic procedural combat tactics." Game Developers Conference. 2005.

[6] Miles, Chris, and Sushil J. Louis. "Towards the co-evolution of influence map tree based strategy game players." Computational Intelligence and Games, 2006 IEEE Symposium on. IEEE, 2006.

[7] Jang, Su-Hyung, and Sung-Bae Cho. "Evolving neural npcs with layered influence map in the real-time simulation game ‘conqueror’." Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On. IEEE, 2008.

[8] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." Systems Science and Cybernetics, IEEE Transactions on 4.2 (1968): 100-107.

[9] Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Numerische mathematik 1.1 (1959): 269-271.

[10] Danielsiek, Holger, et al. "Intelligent moving of groups in real-time strategy games." Computational Intelligence and Games, 2008.

[11] Shi, Yinxuan, and Roger Crawfis. "Optimal cover placement against static enemy positions." FDG. 2013.

[12] Shi, Yinxuan, and Roger Crawfis. "Group tactics utilizing suppression and shelter." Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES), 2014. IEEE, 2014.

[13] Shapiro, Jacob, Jerry Waxman, and Danny Nir. "Level graphs and approximate shortest path algorithms." Networks 22.7 (1992): 691-717.

[14] Sturtevant, Nathan, and Michael Buro. "Partial pathfinding using map abstraction and refinement." AAAI. Vol. 5. 2005.