KICEM Journal of Construction Engineering and Project Management
Online ISSN 2233-9582

www.jcepm.org
http://dx.doi.org/10.6106/JCEPM.2012.2.1.035

Development of an Optimal Trajectory Planning Algorithm for an Automated Pavement Crack Sealer

Hyun-Seok Yoo[1], and Young-Suk Kim[2]

Received October 18, 2011 / Revised February 8, 2012 / Accepted February 13, 2012

Abstract: ***In the last two decades, several tele-operated and machine-vision-assisted systems have been developed in the construction and maintenance area, such as pavement crack sealing, sewer pipe rehabilitation, and excavation. In developing such tele-operated and machine-vision-assisted systems, trajectory plans are very important tasks for the optimal motions of robots whether their environments are structured or unstructured. This paper presents an optimal trajectory planning algorithm used for a machine-vision-assisted automatic pavement crack sealing system. In this paper, the performance of the proposed optimal trajectory planning algorithm is compared with the greedy trajectory plans, which are used in the previously developed pavement crack sealing systems. The comparison is based on the computational cost vs. the overall gains in crack sealing efficiency. Finally, it is concluded that the proposed algorithm plays an important role in the productivity improvement of the developed automatic pavement crack sealing system.***

Keywords: crack sealing, machine vision algorithm, trajectory planning, construction automation

I. INTRODUCTION

A. Background and Objectives

Crack sealing is a preventive road crack repair method that prevents not only further cracking, by delaying the development of road cracks, but also road damages due to freezing, by protecting the road substructure with its waterproof function (Lee, Jeong-Ho et al., 2004). Having recognized the advantages of crack sealing and the risks in road maintenance and repair work, many advanced countries have conducted research on the development of automated crack sealing equipment since the 1990s. South Korea, for its part, developed the automated pavement crack sealer (APCS) in 2001, and the field test of its prototype was successfully conducted in 2004.

The current machine vision system of APCS consists of the crack detection and modeling module and the trajectory planning module. The crack detection and modeling module aims to accurately detect and model the spine of the crack network, and is an essential module that determines the accuracy and quality of APCS. Thanks to the continuous research and development being conducted to improve the recognition rate, this module's completely automated crack detection rate has reached 95.5% (Yoo, Hyun-Seok et al., 2004). To complement the module, the manual mapping and editing functions were added to enable APCS so as to make crack detection and modeling possible in any circumstance.

On the other hand, the trajectory planning algorithm is an algorithm that determines in what order the crack network shall be moved and sealed, and determines the speed and productivity of APCS.

Currently, the most popular trajectory planning of APCS is Kim and Haas's (1998) greedy trajectory planning algorithm. It is not only easy to implement but also offers very efficient path results and works fast. Due to the characteristics of the algorithm, however, it cannot always guarantee the shortest path, and as the number of cracks on the road image increases, the idle distance from the optimal result also tends to increase.

From the viewpoint of the graph theory in computer engineering, the trajectory planning on APCS is a problem in which an optimal result cannot be known until all the trajectories are examined. In general, an optimal result can be obtained if the number of cracks is small, while the number of possible trajectories exponentially increases resulting in tremendous time and cost as the number of cracks on the road image increases. The actual result of the field test using APCS shows, however, that the cracks on the road image are between eight and nine at most (they are usually between one and four). Such result signifies that theoretically, all the trajectories within a certain range can be examined, and therefore, it is possible to develop an optimal trajectory planning algorithm that always guarantees the optimal result.

Furthermore, the recent advancement of computing technology improves the scope that can be handled by the optimal trajectory planning algorithm.

This research aims to develop an optimal trajectory planning algorithm for APCS through effective trajectory

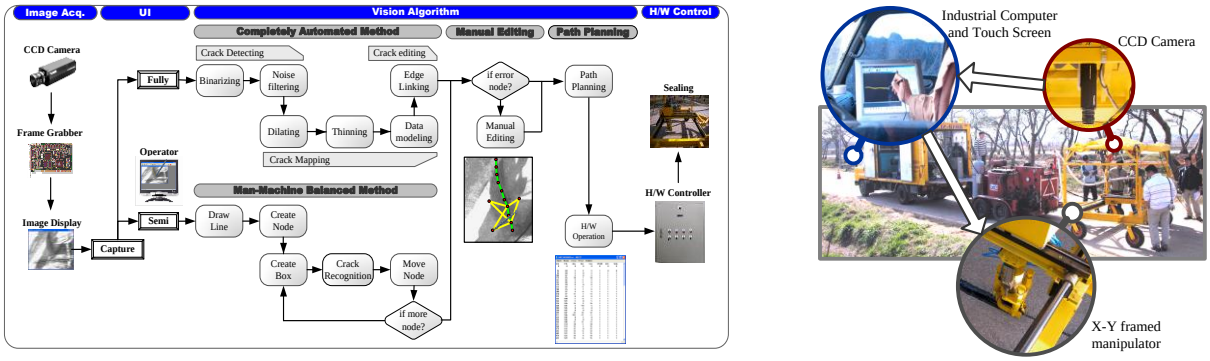
[1] Member and research professor, Ph.D., Department of Architectural Engineering, Inha University, hsyoo.cm@inha.ac.kr
[2] Lifetime member and professor, Ph.D., Department of Architectural Engineering, Inha University, youngsuk@inha.ac.kr

planning data modeling, and to evaluate the degree of improvement in the performance and scope of the developed algorithm compared to the existing algorithm. The result of the research is expected to improve the productivity of APCS.

B. Scope and Methodology

The scope of this research includes the analysis of the problems of the existing trajectory planning algorithm among the machine vision algorithms for controlling APCS, suggesting a new trajectory planning algorithm, and comparing the two algorithms. The methodology of the research includes the following:

(1) to introduce the concept and process of the vision algorithm, which can be used to analyze the role and importance of the trajectory planning algorithm;
(2) to analyze the performance and problems of the previously developed trajectory planning algorithm;
(3) to realize an effective trajectory modeling method and data structure in a crack network that exists in images; and
(4) to analyze the performance and advantages and disadvantages of the existing greedy algorithm and the proposed algorithm, each of which was realized using a programming language, and to suggest an effective application method for APCS.



① Machine vision algorithm ② H/W

FIGURE I
APCS

II. Role Of The Trajectory Planning Algorithm And Survey Of The Previous Algorithm

A. Construction of the Machine Vision Algorithm and Role of the Trajectory Planning Algorithm

The existing vision algorithm, as shown in Figure 1-①, consists of five phases: image acquisition, noise removal, crack detection and mapping, trajectory planning, and crack sealing. In image acquisition, the image of the road surface is acquired by a CCD camera that uses an 8bit gray bitmap with a 640×480 resolution on top of the APCS. The analog images shot by the CCD camera are digitally converted and stored through the frame grabber, and depending on the user's initial zone setting, image processing is performed.

In the crack detection phase, the cracks are distinguished in a binary system from the road images acquired by the CCD camera, through which the noise can be intelligently removed. A previous study (Yoo, Hyun-Seok, 2004) removed noise objects from the road images by successfully using the neural-network learning technology.

The cracking-mapping algorithm, the third phase of the existing algorithm, controls the movement of the sealant injection device so it would move accurately along the center of the crack. It is categorized into full automatic mapping and manual mapping. Full automatic mapping leaves only the skeleton of the crack with the thinning algorithm after the noise is fully removed.

The disadvantage of full automatic mapping is that it works only after the noise has been completely removed. On the other hand, manual mapping is not greatly affected by the remaining noise or the damaged cracks and can accurately extract the center line.

Trajectory planning, the fourth phase, determines the order of movement of the automation equipment's sealant injection device against several extracted cracks. Based on the original point (the end point of the upper left side), the sealant injection device moves one time according to the crack centerline within the image, and if there exist several cracks, the idle distance between the cracks will show a considerable difference according to the crack sealing order. The goal of the trajectory planning algorithm is to detect the network with the shortest idle distance among numerous crack networks.

After trajectory planning, the program combines the crack mapping and trajectory data to output the hardware instructions in text format. These hardware instructions will be transmitted to the equipment control driver based on a protocol, and the sealant injection device will be operated based on these hardware instructions.

The mapping data based on the trajectory generally use array or linked list data structures. What is needed for trajectory planning among the mapping data is the x-y coordinate of both ends (front and rear points) of each trajectory as well as the number of trajectories (hereafter, "N"). In this paper, a structure that combines the array and linked list structures was used. The first front [] and

end [] points of each trajectory are referred to by the array pointer, and the internal trajectory data are linked by the string-type linked list structure. Shown in Figure 2 are the data in which the front and rear points of each trajectory are used in the trajectory planning. Based on n number of trajectory data stored in a queue, the information required for trajectory planning are the x-y coordinates of both the front and rear points of each trajectory. These coordinates are the values that both array points of the queue point out. For example, the starting point of trajectory 3 in Figure 2 has the front[3] → (4,439) coordinate, and its end point coordinate is rear[3]→(508,245). The wood-branch-shaped cracks, which are often detected in crack images, should be separated by string-type linear data and stored. The trajectory planning algorithm has the following principle:

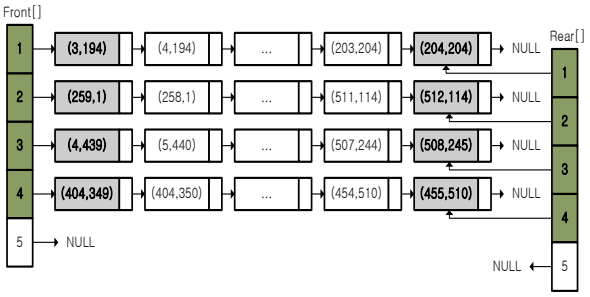


FIGURE II
STORAGE STRUCTURE OF THE CRACK NETWORK MAPPING DATA

(1) The original point of the trajectory planning is located at the end point of the upper left area of the image, and its coordinate is (0,0). The original point is the location at which the sealant injection device of the automation equipment waits for the work order.
(2) All the crack objects are sealed at one time. In other words, once sealed, the crack objects will be excluded from the next trajectory movement.
(3) The sum of the crack sealing distance is identical among various trajectory plans, and therefore, the most efficient trajectory planning is the one whose idle distance is the shortest.
(4) As the automation equipment moves to the next work area after completing the sealing on all the crack objects, the return distance of the injection device to the original location is not included in the idle distance.

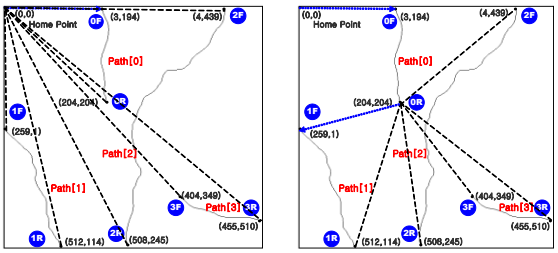
The trajectory planning algorithm is directly related to the work speed and productivity of the automation equipment. If the average speed of the sealant injection device is 150 mm/sec, its work area will be 2.0×1.5 m, and the resolution of the image area will be 640×480. If the average size of one pixel is 3.125 mm and the idle distance between the optimal and worst trajectory plans is 800 pixels, the actual idle distance of the sealant injection device will be 2,500 mm, and each sealing will delay the work time by 16.6 seconds. Considering that the average daily crack sealing workload is 1.4 km, up to two hours of productivity difference per day will be generated only by the trajectory planning algorithm.

B. The Existing Trajectory Planning Algorithms and Problems

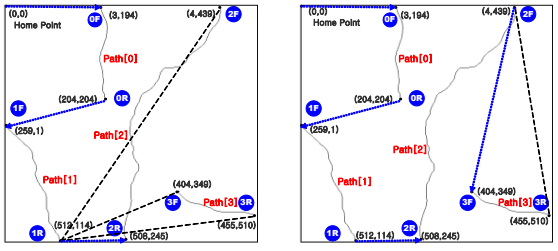
1) Greedy Trajectory Planning Algorithm

Kim and Haas's greedy trajectory planning algorithm (1998) establishes trajectory planning in such a way that if various crack networks exist within a work area, the trajectory will move in sequence from the visited point to the closest crack network based on the following process:
(1) Search for the closest end point among the unvisited cracks.
(2) Move the sealant injection device to the closest end point and seal the crack up to the opposite end point, and once the task is completed, record the crack as the visited crack.
(3) Set the end point opposite the direction to which the sealant injection device moved as the visited point of the search.
(4) Verify if there is any unvisited crack from the visited point of the search, and if there is any, return to phase (1). If there is none, move the sealant injection device to the original point and complete the crack sealing process.

In the greedy trajectory planning algorithm, the original point becomes the first visited point, and once the sealant injection device moves to a point of the closest crack network, the opposite end point of the crack network becomes the next visited point. By repeating such process, the greedy trajectory planning algorithm can plan the trajectory among the cracks. Shown in Figure 3 is an image of four crack networks, for which the process of the greedy trajectory planning algorithm is as follows:
(1) Calculate the distance between both ends of the four crack networks (i.e., eight end points) based on the original point, and move the sealant injection device to the closest end point (0F) (Figure 3-①).
(2) Seal the crack from 0F to 0R, set 0R as the visited point, measure the distance to the other points, and move the sealant injection device to the closest end point (1F) (Figure 3-②).
(3) Seal the crack from 1F to 1R, set 1R as the visited point, measure the distance to the other points, and move the sealant injection device to the closest end point (2R) (Figure 3-③).
(4) Seal the crack from 2R to 2F, set 2F as the visited point, measure the distance to the other points, and move the sealant injection device to the closest end point (3F) (Figure 3-④).
(5) Seal the crack from 3F to 3R and return the sealant injection device to the original point.



① 1st movement ② 2nd movement



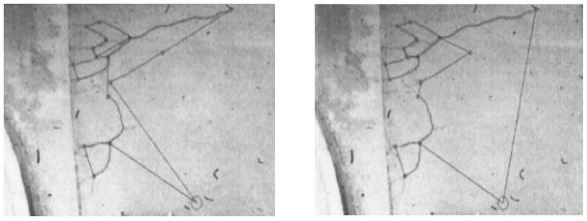
③ 3rd movement ④4th movement

FIGURE III
GREEDY TRAJECTORY PLANNING ALGORITHM PROCESS

Kim and Haas's greedy trajectory planning algorithm allows good paths to be found within a very short time. In fact, the time that it takes for this algorithm to calculate the trajectory for ten or more cracks in one image is 3milliseconds. The result of the calculation of the greedy trajectory planning algorithm, however, as shown in Figure 3, has a considerable difference from the shortest-distance trajectory. The distance calculated by the greedy algorithm is 945 pixels whereas the actual shortest distance (home-1F-2R-0F-3F) is 882 pixels, a difference of 63 pixels (7.1%). Moreover, such a difference tends to increase as the number of cracks (n) increases.

2) Trajectory Planning Using Simulated Annealing

To improve the existing greedy algorithm, Mathurin and Velinsky (2000) developed a trajectory planning algorithm that uses simulated annealing (SAa), a metaheuristic algorithm. It creates a permutation of the crack trajectories, randomly selects a neighboring trajectory, and discovers a shorter trajectory using a probability function. It uses Boltzmann probability distribution to avoid a large amount of calculation, and 0.88 as α vector, the temperature reduction function. According to Feng et al. (2005), the trajectory planning algorithm based on simulated annealing may not always guarantee shortest-trajectory planning, as shown in Figure 4, as it is reported to reduce the idle distance compared to that of the existing greedy algorithm by up to over 15%.



① Greedy algorithm ② Simulated annealing

FIGURE IV
RESULT OF SIMULATED ANNEALING (FENG ET AL., 2005).

III. TRAJECTORY PLANNING ALGORITHM DESIGN

A. Observation of the Shortest-Trajectory Planning Algorithm

The trajectory planning for the sealant injection device is closely related to the traveling salesman problem(TSP), a representative NP-class [1] in algorithm studies that determines the least expensive path for a salesman visits all n number of points (no duplicate visits) and returns to the original place (Figure 5). That is, TSP implies that all the possible paths should be searched in order to find the shortest distance in a given network.

Since TSP must visit all n number of points, the computing time[2] becomes O(n!). While when n is small, not much time is used until the process is completed, if n increases, the required time multiplies by n, resulting in requiring considerable time as n increases. For example, if the time required for three points is 1 millisecond, the time that it would take to find the shortest distance among 20 points is as much as 6.84 million years. Thus, TSP is categorized as an NP class.

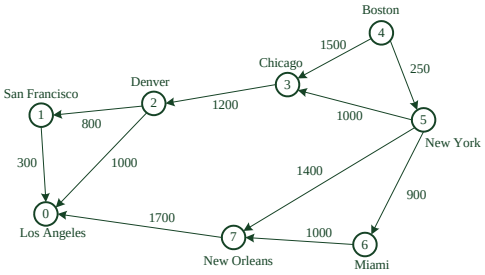


FIGURE V
TSP (HOROWITZ ET AL., 1993).

[1] In 1971, Stephen Cook defined NP class as a category of problems which, if P class is a group of problems that can be solved within a given time and not-P class is a group of problems that cannot be solved within a given time, can be solved within a given time and whose solution can be verified (Decker, 1996).

[2] Asymptotic notation is a method of expressing the comparison of the increase rate of a function with that of another function. It is used to simplify the computing time of an algorithm or the latter part of an infinite series. In defining the computing time of a function, the big O notation, which expresses the worst-case computing time, is widely used. The actual calculation time of an algorithm is measured by millisecond using the clock() function, which is supported by the system in which the algorithm has been implemented.

As has been mentioned, TPS has no developed algorithm and simply searches all paths and finds the optimal one. When the number of places to visit (n) is large, the time cost of TSP is considerable, and therefore, a greedy algorithm used in Dijkstra (1959), Kruskal (1956), Prim (1957), and Sollin (1962) is instead used. A greedy algorithm is a method that chooses for the next path the one with the minimal cost weight value among the points that have yet to be visited. While a greedy algorithm can help find a good solution, it does not guarantee the optimal path. Particularly, as n increases, the probability of finding the optimal path decreases.

The trajectory planning algorithm of APCS is to plan the trajectory order for moving the sealant injection device to the cracks and sealing them. If the number of cracks existing in the image is n, the number of possible trajectory plans is 2n×n!. In other words, if there are eight cracks, 10,321,920 trajectory plans are possible. The optimal trajectory plan is the plan with the shortest trajectory, and there always exist one or more optimal trajectory plans.

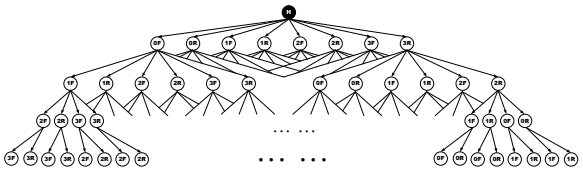


FIGURE VI
TRAJECTORY PLANS FOR APCS.

The trajectory plan for APCS is very similar to that for TSP and is one step more complex than the latter. Whereas the computing time of TSP is O(n!), the trajectory plan for APCS is O(n!×2n). While TSP is simply about visiting points, the trajectory plan for APCS becomes more complex by 2n because one crack (n) of APCS has two vertices. Figure 6 shows the tree-shape intuitive modeling of four cracks in the image, which shows a total of 384 trajectories. In such model, however, the size of the tree increases exponentially as the number of cracks (n) increases. The size of a tree is determined by multiplying the number of nodes (TN) by the size of a node (8 bytes). TN based on n is shown in equation (1). As such, when modeling the whole tree where the number of cracks (n) is eight, the size of the whole tree becomes 136 Mbytes; if n is nine, the size of the tree becomes 2.4 Gbytes, which may exceed the memory capacity of a regular computer.

$$\begin{aligned} TN &= 1 + 2n + 2n \times (n-1) + \cdots + 2^n \times n! \\ &= \frac{2^n \times n!}{2^n \times n!} + \frac{2^n \times n!}{2^{n-1} \times (n-1)!} + \cdots + \frac{2^n \times n!}{1} \\ &= \sum_{i=0}^{n} \frac{2^n \times n!}{2^i \times i!} \end{aligned} \qquad [E.1]$$

As with TSP, the trajectory planning for APCS is an NP class. That is, the optimal path can be known only by visiting all the possible paths. If the computing time of the trajectory plan for APCS is O (n!×2n), however, it can be divided into O(n!) and O(2n). In other words, the problem can be divided into those about the order in which the cracks should be visited (O(n!)) and those about the direction of crack networks that the sealant injection device should approach (O(2n)). This method is a key element of the solution to the trajectory planning for APCS.

B. Two-Phase Tree Algorithm for Searching for the Shortest Trajectory

The optimal trajectory planning algorithm models all the possible paths that pass n number of trajectories based on the original point O(0, 0) by data structure, and selects the trajectory with the shortest idle distance through calculation. The method that this research uses to model the number of all possible trajectories is a two-phase tree structure. If the group of the order in which, based on the original point, all the trajectories pass once is the trajectory group (Table 1), the first tree determines the number of possible cases with regard to the order of each trajectory within the trajectory group, without considering both ends (front[], rear[]) of the cracks. The second tree in two-phase tree algorithm determines the number of cases with regard to the order of points that will enter both ends of each trajectory (front[], rear[]) from the trajectory order acquired in the first tree. From this two-phase tree structure, the number of all possible cases in relation to n number of trajectories can be acquired, and later, the idle distance can be calculated.

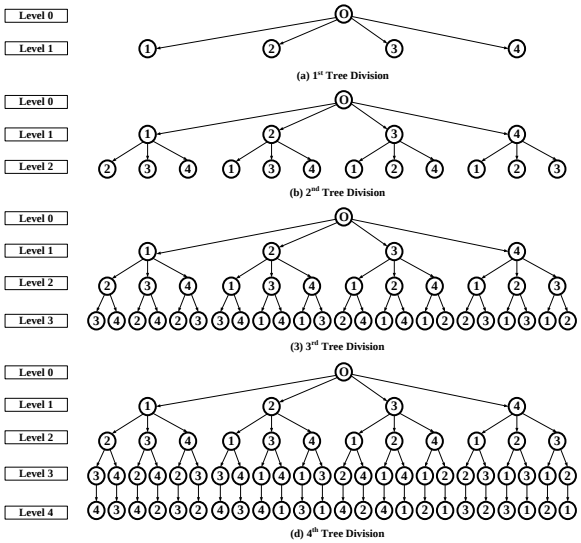


FIGURE VII
THE FIRST TREE TRAJECTORY.

The first tree uses the number of cracks (n). For example, in Figure 7, the number of trajectories is four. Based on this value, the trajectory group ({0, 1, 2, 3, 4}) with five elements (trajectories), which includes the original point (0), is created. The goal of the first tree is to acquire the permutation series based on the order of elements within the trajectory group. First, if the process of selecting and moving to a specific trajectory from the original point is modeled as a tree structure (Figure 7(a)),

the original point (0) is expressed by the highest parent node. Four trajectories to which the device can move from the original point are then expressed as child nodes. In other words, the beginning point is the parent node, and a trajectory to which the sealant injection device can move is expressed as a child node.

In level 2, the four nodes of level 1 become parent nodes, to which the three trajectories that were not passed are expressed as child nodes. Here, the child nodes of one node are identical to the sibling nodes of a parent node. For example, the child nodes whose parent is trajectory 1 node in level 1 in Figure 5(b) are trajectories 2, 3, and 4, save the original point and trajectory 1. Similarly, the sibling nodes of trajectory 1 in level 1 are trajectories 2, 3, and 4. With the same iterative method, the tree for each node can be constructed in levels 3 and 4. Once the construction of the first tree is completed, each trajectory will be traversed to acquire the result of the trajectory group, which is shown in Table 1 in the (trajectory series) format.

The generalization of the number of cases with regard to the order of crack trajectories based on the results of Figure 7 and Table 1 shows that, as in Table 2, the number of trajectory groups whose elements are n trajectories is n!, which is the same as that of the terminal nodes.

TABLE I
TRAJECTORY GROUPS IN PHASE 1 TREE

S Series	Trajectory Group[1]
$[1]^1$	0 → 1 → 2 → 3 → 4
$[2]^1$	0 → 1 → 2 → 4 → 3
$[3]^1$	0 → 1 → 3 → 2 → 4
$[4]^1$	0 → 1 → 3 → 4 → 2
	……
$[21]^1$	0 → 4 → 2 → 1 → 3
$[22]^1$	0 → 4 → 2 → 3 → 1
$[23]^1$	0 → 4 → 3 → 1 → 2
$[24]^1$	0 → 4 → 3 → 2 → 1

TABLE II
NO. OF NODES BY LEVEL

Level (L)	No. of Child Nodes (C)	No. of Nodes (T)
0	n	1
1	n-1	n
2	n-2	n×(n-1)
3	n-3	n×(n-1)×(n-2)
	…	
n-2	2	n×(n-1)×(n-2)×...×3
n-1	1	n×(n-1)×(n-2)×...×3×2
n	0	n×(n-1)×(n-2)×...×3×2×1 = **n!**
	C = n-L	**T = n! / C!** (except when C is 0)

The second tree uses the trajectory groups acquired from the first tree. For example, the trajectories (elements) of the third trajectory group ([3]) are {0, 1, 3, 2, 4}. Each trajectory in the trajectory group has two end points (front[n], rear[n], hereafter F[n] and R[n]). The goal of the second tree is to determine the advancing point with regard to both end points of a trajectory within the given trajectory group.

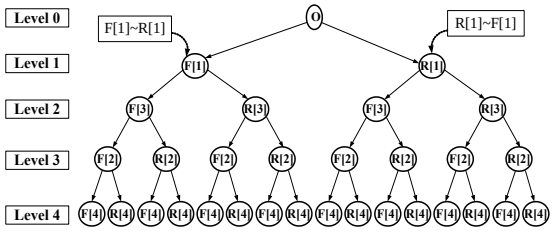


FIGURE VIII
THE SECOND TREE TRAJECTORY

The second tree, based on trajectory group $[3]^1$={0, 1, 2, 3, 4} acquired from first tree, has a complete binary tree, as shown in Figure 8. Each node shown in the second tree pertains to the entry point. For example, node F[1], shown as [F[1]-R[1]] in Figure 8, enters F[1], moves along the crack centerline, and stops at R[1]. Shown in Table 3 is the result of the trajectory groups in the second tree in the form of [trajectory series]^1[trajectory series]^2.

TABLE III
TRAJECTORY GROUPS OF THE SECOND TREE

Level (L)	No. of Child Nodes (C)	Total Node Nos. (T)
0	2	$1 = 2^0$
1	2	$2 = 2^1$
2	2	$4 = 2^2$
3	2	$8 = 2^3$
	…	
n-2	2	2^{n-2}
n-1	2	2^{n-1}
n	2	2^n
	C = 2	$T = 2^L$

When the number of possible cases with regard to the trajectory order to the entry point, based on the results shown in Figure 8 and Table 3, is generalized, the number of possible cases that may occur at one trajectory group while considering the order of entry points becomes 2n (Table 4). In the second tree, the number of terminal node is identical with that of the first tree.

As this is the number of one group {0, 1, 3, 2, 4} among the 24 trajectory groups in the first tree, the number of possible cases of all trajectory groups in the first tree (i.e., the total number of trajectory groups to *n* number of trajectories) becomes $n!\times 2^n$.

TABLE IV
NO. OF NODES BY LEVEL

Series	**Trajectory Group[2]**
$[3]^1[1]^2$	0 → F[1] → F[3] → F[2] → F[4]
$[3]^1[2]^2$	0 → F[1] → F[3] → F[2] → R[4]
$[3]^1[3]^2$	0 → F[1] → F[3] → R[2] → F[4]
$[3]^1[4]^2$	0 → F[1] → F[3] → R[2] → R[4]
	(omit)
$[3]^1[13]^2$	0 → R[1] → R[3] → F[2] → F[4]
$[3]^1[14]^2$	0 → R[1] → R[3] → F[2] → R[4]
$[3]^1[15]^2$	0 → R[1] → R[3] → R[2] → F[4]
$[3]^1[16]^2$	0 → R[1] → R[3] → R[2] → R[4]

The calculation of the idle distance uses the trajectory groups in the second tree as the input values. For example, in Table 3, trajectory group [3]1[14]2 can be expressed by

[3]1[14]2= {0 → R[1] → R[3] → F[2] → R[4]}

= {0 → R[1]～F[1] → R[3]～F[3] →F[2]～R[2] → R[4] ～F[4] },

and since the (→) region is a non-sealing distance, the total idle distance among the cracks is

TOTAL IDLE DISTANCE

= THE DISTANCE BETWEEN ORIGINAL POINT (0) AND R[1]

+ THE DISTANCE BETWEEN F[1] AND R[3]

+ THE DISTANCE BETWEEN F[3] AND F[2]

+ THE DISTANCE BETWEEN R[2] AND R[4]

In the actual programming stage, as shown in Figure 9, the straight-line distance values between these sets of two points are calculated in advance to speed up the calculation, and the result is stored in a two-dimensional sequence, which is called upon during the calculation process. For example, the calculation of the distance between F[1] and R[3] in Table 5 refers to the sequence distance [1][6] to acquire the value of 287.1, resulting in a faster calculation speed.

TABLE V
COST-NEIGHBORING MATRIX TO THE DISTANCE BETWEEN TWO POINTS

Vertex		O	F[1]	R[1]	F[2]	R[2]	F[3]	R[3]	F[4]	R[4]
	Dist.	[0][0]	[1][0]	[2][0]	[3][0]	[4][0]	[5][0]	[6][0]	[7][0]	[8][0]
O	[0][0]	0.0	446.0	341.2	340.4	575.2	245.1	339.1	248.0	529.7
F[1]	[1][0]	446.0	0.0	286.6	288.6	536.4	201.0	287.1	505.4	585.4
R[1]	[2][0]	341.2	286.6	0.0	2.2	288.1	216.5	2.2	257.4	307.4
F[2]	[3][0]	340.4	288.6	2.2	0.0	287.2	217.4	2.0	255.3	305.7
R[2]	[4][0]	575.2	536.4	288.1	287.2	0.0	504.5	289.2	371.2	118.0
F[3]	[5][0]	245.0	201.0	216.5	217.4	504.5	0.0	215.4	342.2	511.9
R[3]	[6][0]	339.1	287.1	2.2	2.0	289.2	215.4	0.0	255.5	307.6
F[4]	[7][0]	248.0	505.4	257.4	255.3	371.2	342.2	255.5	0.0	298.2
R[4]	[8][0]	529.7	585.4	307.4	305.7	118.0	511.9	307.6	298.2	0.0

The total idle distance for trajectory group [3]1[14]2 is 963.7 pixels (=341.2+287.1+217.4+118.0). This is the 42nd result among the total of 384 result values. The optimal trajectory search algorithm calculates the idle distance of each of the n!×2n number of trajectory groups and selects the group with the smallest value, which is actually trajectory group [21]1[6]2={0→F[4]→ R[2]→R[1]→F[3]}, whose idle distance is 569.2 pixels. In the calculation of the idle distances of all the trajectories, the second tree always guarantees the shortest trajectory.

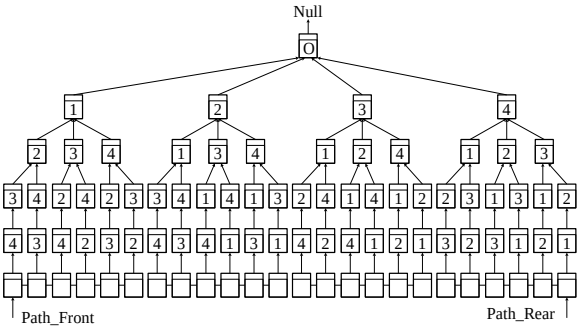


FIGURE IX
DATA STRUCTURE MODEL.

The actual data structure model of the two-phase tree algorithm is a tree structure with reverse pointers, as shown in Figure 9, and under the terminal nodes, a separate pointer queue is created to load the trajectory data one by one. This is because loading the trajectory data from the bottom up is faster and simpler than the other way around. Figure 10 shows the flowchart of the two-phase tree trajectory planning algorithm, and Figure 11 is the text file output of the result from the two-phase tree algorithm, where the number of cracks (n) is 4. Here, H is the original point, and 0-3 are the crack numbers.

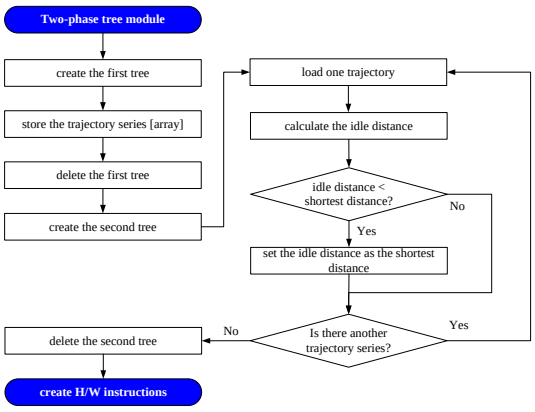


FIGURE X
TWO-PHASE TREE ALGORITHM FLOWCHART.

C. One-Phase Tree Algorithm for Reducing the Time Cost

Although the two-phase tree algorithm guarantees the shortest distance, it requires a very high time cost. In fact, when the number of cracks (n) is 7, the calculation time exceeds 3 seconds, and if n is 8, the calculation time exceeds 72 seconds. To address this problem of the two-phase tree algorithm, this research developed the one-phase tree algorithm that uses only first tree for the trajectory series, and that selects the closest distance for the front and end points of the crack. Without the creation and deletion of the second tree, the one-phase tree algorithm can reduce the total calculation load by half compared to the two-phase algorithm. Shown in Figure 12 is the one-phase trajectory planning algorithm flowchart.

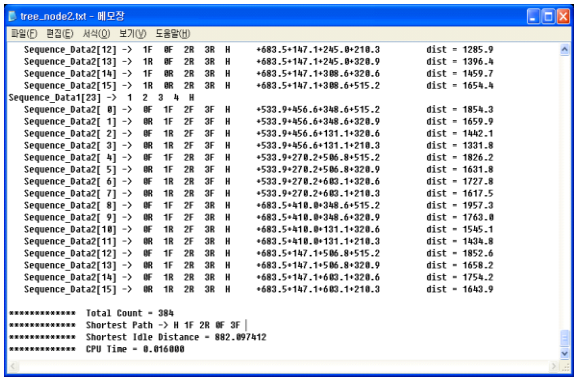


FIGURE XI
TREE EXECUTION RESULT.

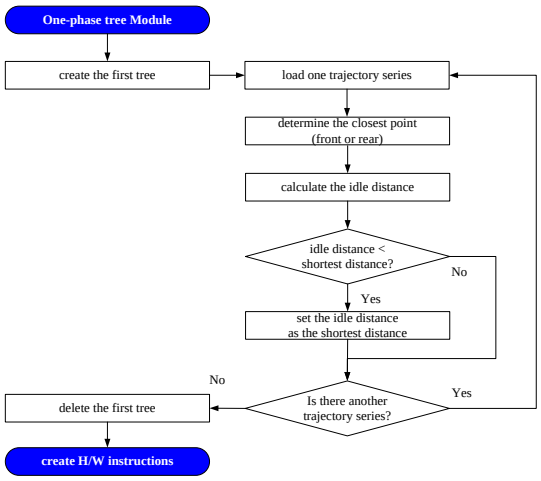


FIGURE XII
ONE-PHASE TREE ALGORITHM FLOWCHART

IV. ANALYSIS OF THE PERFORMANCE OF THE PROPOSED ALGORITHM

A. Environment for the Performance Evaluation of the Trajectory Planning Algorithm

The most important items in evaluating the performance of the proposed algorithm are the calculation time and the idle distance between the cracks, which are directly related to the overall processing time of the vision algorithm. In this study, Microsoft Foundation Class (MFC)'s clock() function was used to measure the calculation time. The clock() function measures time by millisecond. The measurement scope of the calculation time is from the point in time when the trajectory planning function is executed to the point in time when the result is outputted by text. The idle distance between cracks is the coordinate difference between two points when the size of one pixel is considered 1, based on the 640×480 pixel raw image. In other words, the distance between A(x1, y1) and B(x2, y2), D, is shown in the equation below, and the idle distance between cracks is the sum of all the idle distance values that exist within the trajectory.

$$D_{A\sim B} = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \qquad [\text{E. 2}]$$

The data that were used in evaluating the performance of the proposed trajectory planning algorithm were a total of 270 images, each of which was a 640×480 pixel raw image. The number of cracks increased from 1 to 9 in every 30 images. Shown in Table 6 is the environment for the performance measurement of the proposed algorithm.

TABLE VI
SYSTEM SPECIFICATIONS USED IN THE PERFORMANCE MEASUREMENT

CPU	Pentium 4 2.4Ghz(C) Northwood
RAM	DDR 512M (400MHz)
OS	Windows XP (SP1)

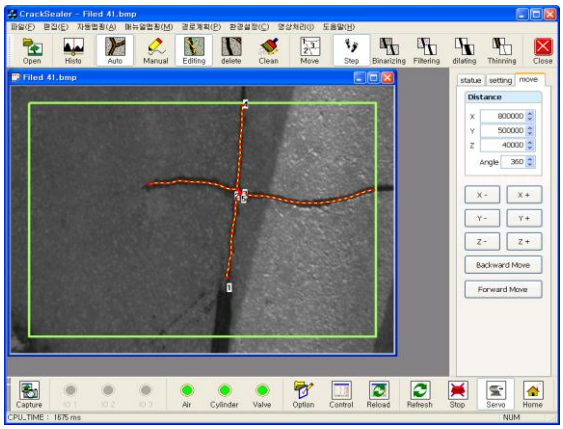


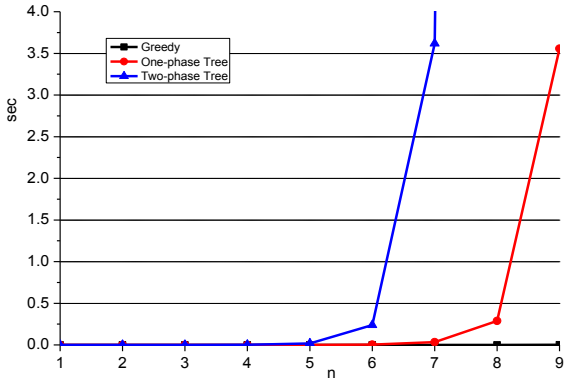
FIGURE XIII
APCS CRACK NETWORK MODELING S/W.

The developed two-phase and one-phase tree algorithms were integrated into the APCS crack network modeling S/W shown in Figure 13, which was realized using Microsoft Visual C++ 6.0. The said S/W has integrated all the functions required for the operation of APCS (e.g., image acquisition, crack detection, crack network modeling, trajectory planning, and motion control).

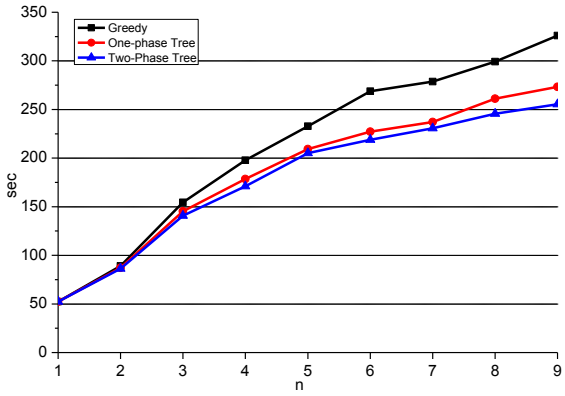
TABLE VII
MEASUREMENT RESULTS OF THE PROPOSED TRAJECTORY PLANNING ALGORITHM

n	Average Calculation Time (Unit: milliseconds)			Average Idle Distance (Unit: pixel)		
	Greedy	One-phase Tree	Two-phase Tree	Greedy	One-phase Tree	Two-phase Tree
1	0	0	0	52.38	52.38	52.38
2	0	0	0	89.08	87.09	86.28
3	0	0	0	154.51	145.26	140.68
4	0	0	2	197.86	178.53	171.03
5	0	0	18	232.76	209.23	205.03
6	0	4	239	268.83	227.12	218.90
7	0	33	3,618	278.63	237.02	230.70
8	0	289	72,843	299.14	261.06	245.67
9	2	3,557	1,767,980	325.94	273.21	255.38

B. Measurement Results of the Performance of the Proposed Trajectory Planning Algorithm



① Calculation time of the proposed trajectory planning algorithm



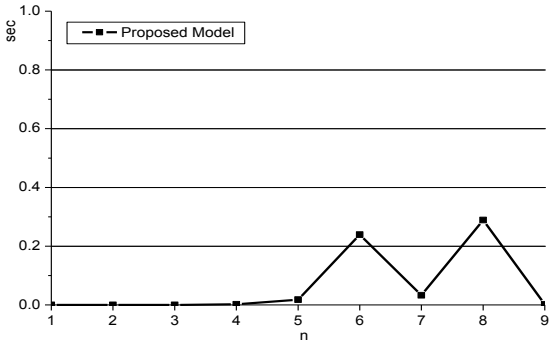
② Idle distance of the proposed trajectory planning algorithm

FIGURE XIV
CALCULATION TIME AND IDLE DISTANCE OF THE PROPOSED TRAJECTORY PLANNING ALGORITHM.

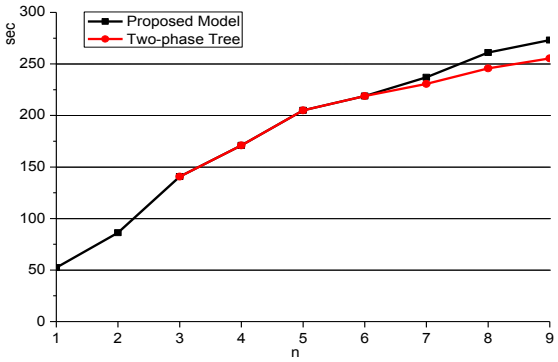
Table 7 shows the results of the measurement of the calculation time and idle distance of the existing greedy algorithm and the developed two-phase and one-phase tree algorithms. While the calculation time of the greedy algorithm was 2 miliseconds or less at most despite the increase in the number of cracks (n) to 9, that of the one-phase tree trajectory planning algorithm was 3,557 milliseconds on average when n was 9, exceeding 1 seconds.

Also, the calculation time of the two-phase tree trajectory planning algorithm was 3,618 milliseconds on average when n was 7, exceeding 1 second. As shown in ② in Figure 14, compared to the two-phase tree trajectory planning algorithm that guarantees the optimal result, the greedy algorithm found a more distant trajectory by an average of 18.24%, and as n increased, the difference also increased. Meanwhile, the one-phase tree trajectory planning algorithm found a more distant trajectory by 4.03% on average compared to the two-phase tree trajectory planning algorithm. As for the greedy algorithm, as n increased, the difference also increased.

C. Application of the Trajectory Planning Algorithm for APCS



① Calculation time of proposed model



② Idle distance of proposed model

FIGURE XV
TRAJECTORY PLANNING MODEL OF APCS.

As has been mentioned, the two-phase and one-phase tree trajectory planning algorithms have a limited scope of application according to the number of cracks, and therefore, it was ensured in this research that the proposed algorithms for APCS appropriate to the number of cracks were to be executed. In other words, if n was between 1 and 6, the two-phase tree algorithm was executed, and if n was between 7 and 8, the one-phase tree algorithm was executed. If n was 9 or over, the greedy trajectory planning algorithm was executed. As a result, as shown in Figure 15, the trajectory planning model completed its process within up to 0.3 seconds, and the idle distance also showed an average difference of 5.7% compared to the optimal value.

V. CONCLUSION

A trajectory planning algorithm uses automatically recognized crack trajectory information from APCS to find the shortest trajectory for the sealant injection device of APCS, and to perform sealing fast and efficiently. In this research, two-phase and one-phase tree algorithms were developed, and their performances were compared to that of the existing greedy algorithm. The following conclusions can be drawn based on the study results that were obtained:

(1) In this study, the role of the trajectory planning algorithm, among the machine vision algorithms for APCS, was analyzed, and the concept of the trajectory planning algorithm and the change in productivity

according to the difference in performance were examined.

(2) The results of the analysis of the existing greedy trajectory planning algorithm showed that while it is a very fast and effective algorithm, its accuracy was decreased as the number of cracks increased.

(3) Through the theoretical observation of the optimal trajectory planning algorithm, it was determined that a trajectory planning algorithm for APCS could be divided into the first tree, which is about the order of trajectories, and the second tree, which is about the order of entry, and could be modeled accordingly.

(4) Through the first tree structure modeling with regard to the order of trajectories, a trajectory series was created. By modeling the second tree structure to determine the order of entry, the two-phase tree trajectory planning algorithm which calculates all the passes for n, the number of cracks was then developed.

(5) Furthermore, to reduce the time cost of the two-phase tree algorithm, one-phase tree trajectory planning algorithm, which creates the trajectory series with the first tree and selects the closest location for the order of crack entry, was developed.

(6) The existing greedy algorithm and the developed two-phase and one-phase tree algorithms were developed as S/Ws. Using the 270 images of trajectory data, the calculation time and the accuracy of each algorithm were tested. The results showed that while the greedy algorithm was the fastest, it was the least accurate (average difference: 18.24%), and that the two-phase tree algorithm was the slowest but always produced the optimal trajectory. The one-phase tree algorithm was faster than the two-phase tree algorithm, and its average difference was almost close to that of the two-phase tree algorithm (4.03%).

(7) To have an appropriate algorithm executed according to n, the number of cracks, the conditions were set so that when n was between 1 and 6, the two-phase tree algorithm was to be executed, and when n was between 7 and 8, and 9 or over, the one-phase tree algorithm or the greedy trajectory planning algorithm was to be executed, respectively. As a result, the trajectory planning model of APCS could perform its process within up to 0.3 seconds, and its idle distance had the average difference of 5.7% compared to the optimal value.

Finally, it is concluded that the proposed one-phase and two-phase tree trajectory planning algorithms could contribute to the improvement of overall productivity of APCS. It is also expected that the time and cost performance of the proposed algorithms would be much improved, if other optimization algorithms such as genetic or 2-opt are applied to the optimal trajectory planning of APCS

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2011-0027609).
This research was supported by Inha University.

REFERENCES

[1] Y.S. Kim, “Development and Implementation of an Automated Pavement Crack Sealer”, Final Report by Korea Institute of Construction and Transportation Technology Evaluation and Planning, Ministry of Construction and Transportation, pp. 100-114, 2004.

[2] H.S. Yoo, J.H. Lee, Y.S. Kim, J.R. Kim, “The Development of a Machine Vision Algorithm for Automation of Pavement Crack Sealing”, *Korean Journal of Construction Engineering and Management*, KICEM, vol. 5, no. 2, pp. 90-104, 2004.

[3] H.S. Yoo, J.H. Lee, Y.S. Kim, N.W. Sung, “A Study on the Development of Pavement Crack Recognition Algorithm Using Artificial Neural Network,”, Proceedings of the 2004 Korea Institute of Construction Engineering and Management, pp. 561-565, 2004.

[4] J.H. Lee, H.S. Yoo, Y.S. Kim, M.Y. Cho, “A Study on the Development of an Automated Pavement Crack Sealer”, *Korean Journal of Construction Engineering and Management*, KICEM, vol. 5, no. 2, pp. 162-171, 2004.

[5] E.W. Dijkstra, “A Note on Two Problems in Connection with Graphs", *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[6] X. Feng, R. Mathurin, S.A. Velinsky, “Practical, Interactive, and Object-Oriented Machine Vision for Highway Crack Sealing", *Journal of Transportation Engineering*, vol. 131, no. 6, pp. 451-459, June 2005.

[7] F. Glover, A.P. Punnen, “The traveling salesman problem : New solvable cases and linkages with the development of approximation algorithms", *Journal of the Operational Research Society*, vol. 48, no.5, pp. 502-510, May 1997.

[8] R. Graham, P. Hell, P. "On the history of the minimum spanning tree problem", *Annals of the History of Computing*, vol. 7, no. 1, pp.43-57, 1985.

[9] E. Horowitz, S. Sahni, S. Anderson-Freed, “Fundamentals of Data Structures in C”, W. H. Freeman and Company, 1993.

[10] Y.S. Kim, C.T. Hass, “Path Planning for Machine Vision Assisted, Teleoperated Pavement Crack Sealer”, *Journal of Transportation Engineering*, vol. 124, no. 2, pp. 137-143, March 1998.

[11] K.R. Kirschke, S.A. Velinsky, “Histogram-Based Approach for Automated Pavement-Crack Sensing”, *Journal of Transportation Engineering*, vol. 118, no. 5, pp. 700-710, September 1992

[12] J.B. Kruskal, “On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem”, *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48-50, 1956.

[13] R. Mathurin, S.A. Velinsky, “Simulated Annealing for the Optimal Trajectory Planning of an Automated Crack Sealing Machine”, Proceedings ASME design technical conference, 2000.

[14] R.C. Prim, “Shortest Connection Networks and Some Generalizations”, *Bell System Technology Journal*, vol. 36, pp. 1389-1401, 1957.

[15] G. Sollin, “Problemes de Recherche Operationelle Report”, C. 41 S.E.G. Paris Le Trace des Canalizations, pp. 15-23, 1962.