BUILDING UP HIGH QUALITY ROADMAP USING VERTEX CLASSIFICATION BASED INCREMENTAL PATH PLANNER

YUEQIAO LI[1], DAYOU LI[1], CARSTEN MAPLE[1], YONG YUE[1], ZUOBIN WANG[2]

Abstract. Poor quality roadmaps can lead to in-optimal paths and lengthy path planning processes. Incremental path planners allow undesired vertices of the newly developed components to be removed when updating the existing roadmaps with the new components. Hence, it is possible for the planners to produce high quality roadmaps if quality measures are used to identify the undesired vertices. The usefulness of a vertex in the sense of high quality roadmap construction depends on the type of the region it resides, given a configuration space. Therefore, it is important for such path planners to identify the region types. This paper presents the concept of the usefulness and a vertex-based classification approach to enable the planners to identify the region types and to calculate the usefulness of vertices accordingly.

Key words: robot path planning, quality of roadmap, incremental path planning, region classification.

1. INTRODUCTION

Sampling-based path planners normally involve the construction of roadmaps that will be repeatedly used in the later stage to answer various path planning queries. The process of building up a roadmap can be time-consuming given the complexity of a configuration space. For this reason, the planners often focus on the speed rather than the quality. The quality of a roadmap can be measure through roadmap size, coverage, connectivity and useful circles, among many other criteria. Poor quality roadmaps can have redundant vertices and edges, which does not only lead to high costs in terms of searching time for a path in the roadmap but also to detours in the obtained paths. Some of them can even contain disconnected sub-roadmaps, which does not represent the real connectivity of the environment and causes failures in finding paths.

Incremental path planners, on the other hand, do not require the pre-processing stage of constructing roadmaps. Instead, the planners develop roadmaps incrementally while answering queries. For this reason, speed does not

[1] University of Bedfordshire, Faculty of Creative Arts, Technologies and Science, Department of Computer Science, Park Square, Luton, LU1 3JU, UK

[2] Changchun University of Science and Technology, Centre of Nano Metrology and Manufacturing Technologies, China

Rom. J. Techn. Sci. – Appl. Mechanics, Vol. 57, N^{os} 2–3 P. 169–195, Bucharest, 2012

need to be emphasised in incremental path planners at the same level as in the sampling-based planners and there are more rooms to address the quality issue. Most incremental path planners contain the stage of removing some vertices from the newly developed roadmap components. The quality issue can be emphasised in the way of introducing roadmap quality measures into this stage to calculate the usefulness to the construction of a high quality roadmap for all the recorded vertices. Vertices that hold low level of usefulness will then be identified and removed. Because only the highly useful vertices are retained, the pruned new components are of high quality.

The usefulness of a vertex depends on the type of the region it resides in a C-space. C-space regions can be classified into four types, namely, free region, clustered region, narrow passage and blocked region (Dale and Amato, 2001). The number and the relative location of vertices can mean differently when they are located in different regions. For example, the same number of vertices will be considered less representing in a free region than in a clustered one. That is, they are less useful to constructing a high quality roadmap in the free region. Therefore, it is important to identify the region types when calculating the usefulness of the vertices.

Dale and Amato (2001) also proposed five different features for the types of regions they suggested and a feature-based classification algorithm to identify the regions. Morales *et al.* (2005) suggested a C-space subdivision method. Given a set of samples of a C-space, this method recursively performs partitioning until the features of the obtained regions match those of pre-defined region types. Denny et al (2010) investigated three clustering algorithms for region identification. They evaluated the algorithms on the impact on the quality of the roadmaps developed based on the regions classified using the algorithms.

It worth to notice that all the research mentioned above aims to break a C-space into a number of small regions allowing suitable sampling-based path planners to be applied to the small regions according to the types of the regions. Roadmaps can be built up for the small regions relatively more quickly and of good quality. For the purpose of subdivision, all the algorithms are required to explore the entire C-space to gain the global knowledge of the C-space. However, incremental path planners only explore local workspaces that are related to the queries they are answering.

This paper presents a vertex-based classifier for incremental planners to identify region types. RRT_extension function (LaValle, 1996) and RRT_connection function (Kuffiner and LaValle, 2000) are employed to collect local information about obstacle distribution in the region around a recorded vertex. The information is then stored in the vertex. Vertex types are defined based on this information. Each of these types reflects one of the region types. Features of the vertex types are also defined to enable the classification of vertex types when give the information stored in a vertex and its neighbouring vertices. The planners

can then calculate the usefulness of the vertices recorded according to the recognised region types. The proposed classifier has the advantage of utilising local knowledge rather than global knowledge of a C-space. Therefore, the process of exploring the entire C-space is not required.

The rest of this paper is organised as the following. Section 2 gives related work in the area of roadmap quality, incremental path planners and region type classification. Section 3 presents the proposed vertex-based classifier, including vertex types definition, features of vertex types and classification algorithm. Section 4 introduces the concept of usefulness of vertices in terms of roadmap quality. Section 5 is about the analysis of simulation results. Finally, conclusions are given in Section 6.

2. RELATED WORK

2.1. QUALITY MEASURES

Given a C_{free}, the configurations that are free of obstacles of a C-space, a number of different roadmaps can be constructed with various quality levels. The quality of a roadmap refers to the usefulness of the roadmap to answer queries and the cost of constructing it. Four criteria, namely, the C_{free} coverage ratio, connectivity, useful cycles and roadmap size, are often employed to evaluate the quality of roadmaps.

C_{free} coverage ratio

The prerequisite of making use of a roadmap to answer a query $Q(q_i, q_g)$ is that q_i and q_g can be connected to the roadmap directly without colliding into any obstacle. The collision-free connection of q_i and q_g to the roadmap depends on whether q_i and q_g are in the visibility domain of the roadmap. The visibility domain of a roadmap is the union of visibility domains of all vertices in the roadmap (Siméon *et al.* 2000). The visibility domain $Vis(q)$ of a configuration q, which is illustrated as the shaded area with grid pattern in Fig. 1, is a set of configurations (q') in C_{free} and every path segment $S(q, q')$ is collision-free. The visibility domain $\mathrm{Vis}(q)$ is formulated as:

$$\mathrm{Vis}(q) = \left\{ q' \mid q' \in C_{free}, S(q,q') \subset C_{free} \right\}. \tag{1}$$

The visibility domain $\mathrm{Vis}(R)$ of a roadmap R such that $R = (V, E)$ is the union of the visibility domain of every vertex in the roadmap and is formulated as:

$$\mathrm{Vis}(R) = \bigcup(\mathrm{Vis}(R) \mid v_i \in V). \tag{2}$$

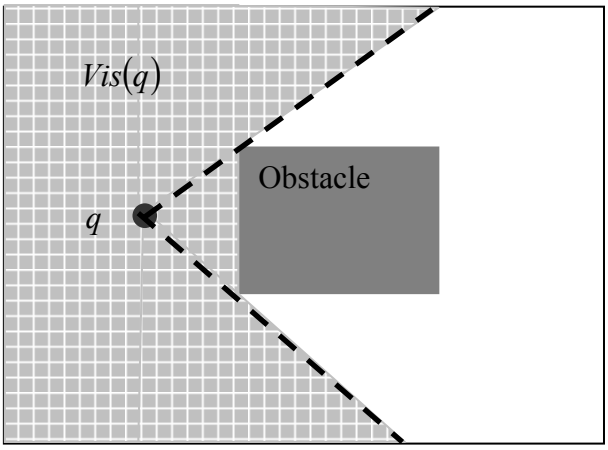

Fig. 1 – The visibility domain of a configuration.

The area of the visibility domain of a roadmap is called the covered C_{free} area of the roadmap. Coverage ratio is the percentage of covered C_{free} by a roadmap in the entire C_{free}. The coverage ratio (CCR) is defined as

$$\mathrm{CCR} = \frac{\mathrm{Vol}_{\mathrm{covered}}(R)}{\mathrm{Vol}_{C_{\mathrm{free}}}(R)}, \tag{3}$$

where $\mathrm{Vol}_{\mathrm{covered}}(R)$ is the volume of covered C_{free} area by a roadmap R and $\mathrm{Vol}_{C_{free}}(R)$ is the volume of the C_{free}, which is represented by R.

A CCR can take a value in the range of [0, 1]. The higher the CCR is, the more area in a C_{free} is covered by a roadmap, R. If a roadmap fully covers a C_{free}, the CCR takes the value of 1. This is called total coverage (Geraerts, 2006). Total coverage means that the visibility domain of a roadmap is C_{free}. Therefore, the total coverage of a roadmap ensures that each configuration q such that $q \in C_{free}$ can be connected to at least one vertex of the roadmap without colliding with any obstacle.

Calculating CCR of a roadmap based on Equation 3 requires the computation of the area of the visibility domain of every vertex in the roadmap, which involves complex geometric calculation and is time-consuming. Geraerts (2006) suggested an easier, approximate calculation method. This method consists of dividing a

C-space into a grid of equal-sized square cells. The finer the C-space is divided, the more exact the CCR obtained using this method. If a configuration q in the middle of a cell is collision-free, the cell is considered to belong to C_{free} and is called a C_{free} cell. If q is visible from at least one vertex of a roadmap, the cell is called visible C_{free} cell for the roadmap. Let $N_{vis}(R)$ be the number of all visible C_{free} cells for a roadmap R and $N_{C_{free}}(R)$ be the number of all C_{free} cells in the C_{free} which represented by R. Then, the CCR of R can be calculated by using the following formula:

$$\mathrm{CCR} = \frac{N_{vis}(R)}{N_{C_{free}}(R)} \tag{4}$$

with a higher CCR, any given pair of connectable q_i and q_g is much more likely to be connected to a roadmap. That is, the roadmap is more useful to answer queries. Improving CCR is a key issue in the improvement of the quality of a roadmap.

Connectivity. Connectivity of a roadmap refers to the vertices of the roadmap are connected. If all vertices are connected, as shown in Fig. 2, the roadmap achieves the maximal connectivity. If a roadmap consists several mini-roadmaps each of which represent an isolated part of an un-connected C_{free}, as illustrated in Fig. 3, and each mini-roadmap is a connected roadmap, this roadmap also achieves the maximal connectivity. If a roadmap representing a connected C_{free} contains fewer mini-roadmaps, then its connectivity is higher than the one that have more mine-roadmaps.

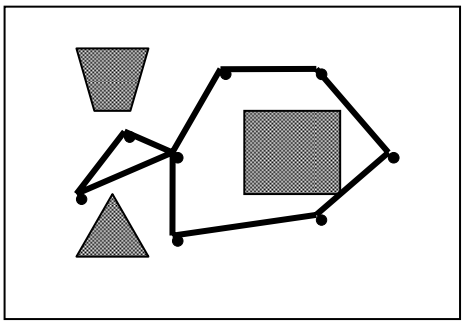

Fig. 2 – A maximally connected roadmap representing a connected C_{free}.

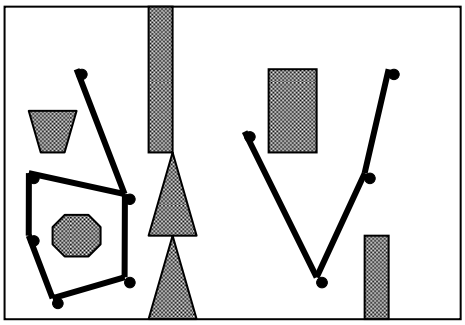

Fig. 3 – A maximally connected roadmap representing an un-connected C_{free}.

Given a roadmap, for a connectable query $Q(q_i, q_g)$ of which q_i and q_g are in the visibility domain, the success of finding a path in the roadmap depends on the connectivity of the roadmap. The maximal connectivity of a roadmap guarantees a path for any connectable query $Q(q_i, q_g)$, provided that q_i and q_g are in the visibility domain of the roadmap. On the other hand, a roadmap with poor connectivity cannot ensure a path even if the roadmap achieves total coverage.

Useful circles. Given a connectable query $Q(q_i, q_g)$, a roadmap is expected to provide the shortest path. This requires two configurations are connected with alternative paths. The alternative paths connecting the same two configurations form a circle in the roadmap. Schmitzberger (2002), Nieuwenhuisen and Overmars (2004), and Jaillet and Siméon (2008) suggested three types of useful circles, namely, un-reducible circles, un-convertible circles and non-first-order deformation circles.

a) Un-reducible circles. A homotopy preserving probabilistic roadmaps (HPPR) provides an exhaustive list of all paths, which are not in the same homotopic class, to answer queries. Paths in different homotopic classes cannot be continuously distorted into each other without colliding with any obstacles. All such paths connecting the same two configurations form un-reducible circles (Schmitzberger, 2002). Figure 4 illustrates such an un-reducible circle. Both paths P_1 and P_2 connect the two same configurations q_i and q_g. P_1, shown with solid lines, cannot be distorted into P_2, as shown with dashed lines, while staying collision-free. The path loop constructed by P_1 and P_2 is an un-reducible cycle.

b) Un-convertible circles. Nieuwenhuisen and Overmars (2004) furthered research on the useful circles in the same homotopic class. If two paths that connect the same two configurations and are in the same hometopic class cannot be distorted into each other by a series of simple smoothing steps without colliding with obstacles, the two paths are said to be un-convertible. Un-convertible paths form an un-convertible circle.

For example, in Fig. 5, paths P_1 and P_3 represented by solid lines to connect q_i and q_g are un-convertible and construct a useful cycle. Since the path P_1 can be distorted into the path P_2 by simple smoothing steps, the cycle constructed by P_1 and P_2 is not a useful cycles.

c) Non-first-order deformation circles. Path deformation roadmaps (PDRs) were proposed by Jaillet and Siméon (2008). PDRs are able to add additional circles based on the measure of the usefulness of a circle in terms of K-order deformation. A *K*-order deformation circle is constructed by two paths P_1 and P_2 in the same homotopic class that can be connected by K ruled surfaces. A ruled surface is a surface that can be swept by moving a straight line in a space. Figure 6

illustrates a first-order deformation circle constructed by two paths P_1 and P_2 which can be connected by a ruled surface. Figure 7 gives a second-order deformation circle constructed by two paths P_1 and P_2 which can be connected by two ruled surfaces. Non-first-order deformation circles are useful ones.

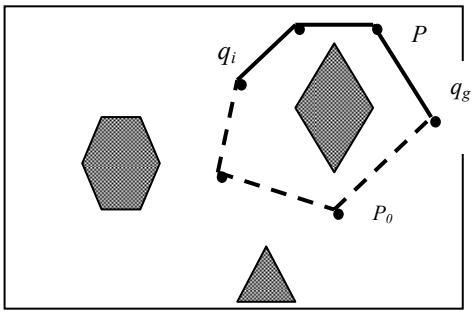

Fig. 4 – An un-reducible circles.

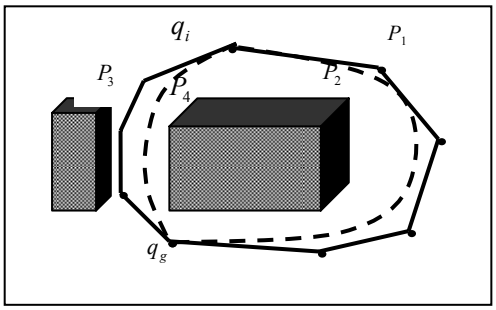

Fig. 5 – Convertible paths and un-convertible paths.

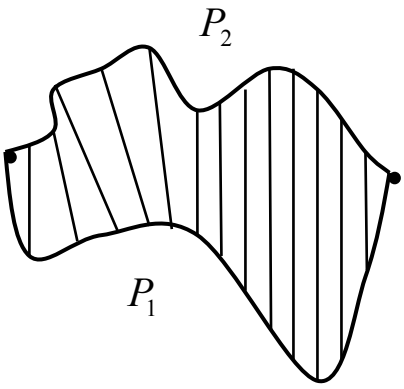

Fig. 6 – A first-order deformation circle.

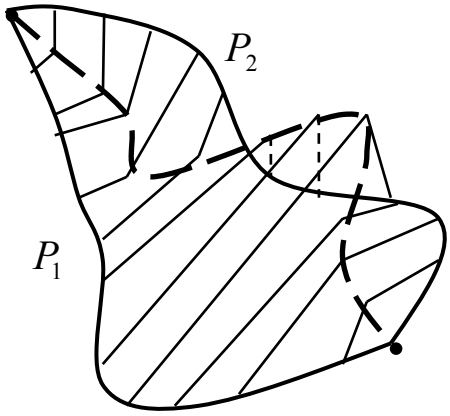

Fig. 7 – second-order deformation circle.

Roadmap size. Roadmap size is the number of vertices in a roadmap. A smaller roadmap consumes less cost in terms of construction and answering queries. Roadmap size sometimes contradicts CCR. This is because with a small number of vertices, it is hard for a roadmap to achieve full coverage.

2.2. INCREMENTAL PATH PLANNERS

Li and Shie (2002) proposed a two-phase RRF algorithm for incrementally building up roadmaps. First, the algorithm answers path queries by growing two

RRTs (LaValle, 1998). After having successfully answered a query, the algorithm records the two newly grown RRTs and merges them with those that were recorded when answering previous queries. During the course of answering more queries, the recorded RRTs gradually form a forest. At the second phase, the algorithm applies pruning to the obtained forest to remove some vertices and edges for the purpose of reducing the size of the forest.

Adorno and Borges (2009) developed an ARW-based approach to the incremental roadmap development. Given a path query, this approach starts two random walks from the starting and the destination configurations, respectively, and saves the successful path as a new component of a roadmap. Similar to the RRF algorithm, the ARW-based approach also involves a process of retaining some vertices in the new component before saving it to the roadmap. In addition, it takes into account of the global connectivity of a given configuration space by choosing the samples from less explored regions among a group of multiple candidates to ensure the random walk covers those regions.

Also aiming to the global connectivity, in particular, in narrow passage regions, of a given configuration space, Kazemi *et al.* (2005) proposed an approach of iteratively updating the existing roadmaps by using topology information about the space obtained through harmonic functions. Given a query, this approach first randomly generates a set of nodes and connects them to form an initial roadmap. It then calculates the velocity of an imaging fluid for each node. As the velocity is high in narrow passage regions, the nodes with high values of velocity are selected as "milestone nodes". A pre-defined number of nodes are then randomly generated within the neighbourhood of each milestone node, so that the distribution of the random nodes is biased towards the narrow passage regions. After a collision detection, the collision-free ones are added to the initial roadmap.

2.3. REGION CLASSIFICATION

Dale and Amato (2002) suggested the following four types of regions for a C-space.

Free region – a free region contains no obstacles. A robot is able to move freely in such a region. A roadmap with simple structure and small number of vertices will be able to represent a free region. There is no need to keep a large number of vertices in the roadmap. For example regions A and E in Fig. 8 are free regions.

Cluttered region – a cluttered region is cluttered with obstacles. The difficulties of constructing a high quality roadmap depends on how cluttered the obstacles resident in the region. For example, regions C and F in Fig. 8 are clustered regions. It is much more difficult to construct a high quality roadmap in region C.

Narrow passage – a narrow passage region contains a narrow passage between other types of regions. It is not easy to construct a path to go throw this region. Therefore, it is essential to keep the path in this region. For example, in Fig. 8, region B is a narrow message region

Blocked region – a blocked region is totally surrounded by obstacles without any path to go to other regions. For example, region D in Fig. 8 is a blocked region.

Although some quality issues such as roadmap size and global connectivity have been taken into account in the above mentioned work, the driving force of the research was to bypass the lengthy pre-processing stage of roadmap construction in sampling-based path planners. Other quality issues, including coverage (Simeon *et al.*, 2000) and useful circles (Schmitzberger, 2002; Nieuwenhuisen and Overmars, 2004; Jaillet and Siméon, 2008), have not been considered in the research. Quality can be emphasised in the way of introducing roadmap quality measures in the process of removing some vertices from the recorded new components of the incremental path planners to calculate the usefulness to the construction of a high quality roadmap for all the recorded vertices. Vertices that hold low level of usefulness will then be identified and removed. Because only the highly useful vertices are retained, the pruned new components are of high quality.

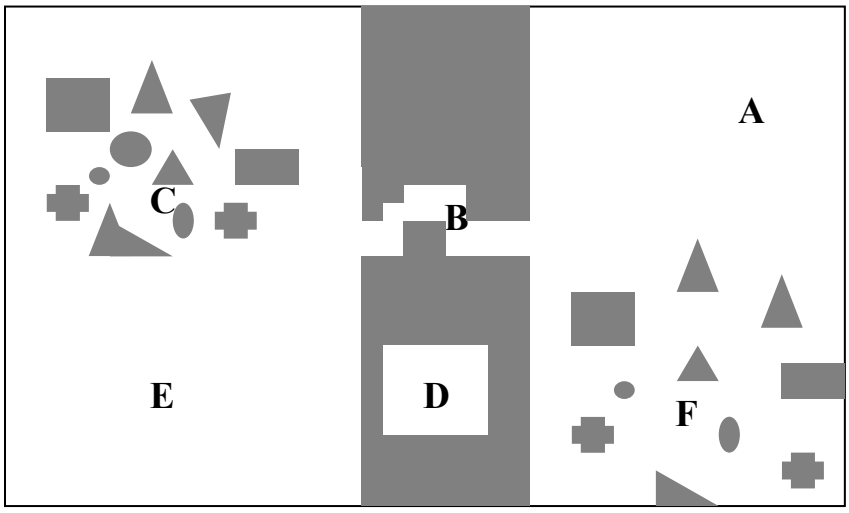

Fig. 8 – Workspace region classification.

3. VERTEX CLASSIFICATION

Vertex type classification aims to identify the types of regions where vertices reside. Therefore, vertex types are meant to reflect C-space region types. Features

of the vertex types are needed so that classification can be carried out based on the features.

3.1. VERTEX TYPES

LaVella (1996) suggested an RRT_extension function which checks whether there is any collision-free path segment to connect an existing vertex v to a new vertex v' along a certain direction in a roadmap. The length of the path segment is a user-defined distance, called extend step. If there is such a short path segment, the extension is successful. The failure to find such a path segment means that v is near to at least one obstacle in the direction. The function returns information about whether an attempt of extension succeeds or fails. Kuffiner and laVella (2000) published another function called RRT_connection function. This function checks if there is a collision-free path segment between two existing vertices, v and v'. The function also tries to find out where is the nearest obstacle along the direction from v to v' if v and v' cannot be connected. A new vertex v'' will be added just before the obstacle. The function returns information about whether v and v' can be connected successfully and also the location of v'', called successful connection distance, if it is added.

The information given by the RRT_extension function indicates whether there is an obstacle along a certain direction and in a certain range from a vertex. The nearest obstacle to a vertex can even be allocated by the RRT_connection function. Repeatedly executing the two functions to a number of different directions will produce rich topology information of the region around a vertex. The information forms a basis for defining vertex types and features of each vertex type. The obtained information is stored in the corresponding vertex.

Four different vertex types are defined according to the topology information, namely, free vertex, near-obstacle vertex, narrow-passage vertex and cluttered vertex.

Definition 1 (Free vertex). A vertex v is a free vertex if $d_1 > D$, where d_1 is the distance from v to the nearest obstacle and D is the successful connection distance.

Vertex v_a in Fig. 1 is a free vertex as no obstacle is located within the range specified by D. The region where a roadmap is constructed by free vertices is free of obstacles.

Definition 2 (Near-obstacle vertex). A vertex v is a near-obstacle vertex if $d_1 \le D$ and $d_2 \le D$, where d_2 is the distance from v to the nearest free vertex.

A near-obstacle vertex normally faces obstacle or obstacles in one or a few directions but nearby free vertices in other directions within the range specified by

D. Vertex v_b in Fig. 1 is a near-obstacle vertex because on one hand it closes to an obstacle and on the other hand it can be connected directly to the nearest free vertex which is located within the range specified by D.

Definition 3 (Cluttered vertex). A vertex v is a cluttered vertex if $d_1 \leq D$, in the cases where there is no any other vertex is near to v, or $d_1 \leq D$ and $d_3 \leq D$, where d_3 is the distance from v to another non-free vertex, otherwise.

Different from the near-obstacle vertices, a cluttered vertex is surrounded by obstacles and can only be connected with other non-free vertex within the range specified by D. Vertex v_c in Figure 1 is such a vertex.

Definition 4 (Narrow-message vertex). A vertex v is a narrow-passage vertex iff $d_1 \leq D$ in two opposite directions, and the nearby vertex v' also share the same property.

Vertices v_d and v_e are narrow-passage vertices in Fig. 1.

3.2. FEATURES OF ROADMAP VERTICES

Features of a roadmap vertex, v, contain its own features and collective ones that are related to its neighbouring vertices. The neighbouring vertices of v are those that are located within a circle centred at v and with the radius of d and are either directly or indirectly connected to v. The reason why the collective features are needed is that sometimes the type of a vertex cannot be classified with its own features alone. All the features can be extracted from the information returned by the RRT_extension and the RRT_connection functions.

Definition 5 (Failed extension ratio of a vertex). For a vertex, v, its failed extension ratio, $R_{fExt}(v)$, is the ratio of the number of failed extensions from v and the total number of extensions made from v, such as

$$R_{fExt}(v) = \frac{N_{fExt}(v)}{N_{fExt}(v)+N_{sExt}(v)}, \tag{5}$$

where $N_{fExt}(v)$ is the number of failed extensions from v and $N_{sExt}(v)$ stands for the number of succeeded extensions from v. $R_{fExt}(v)$ takes values from [0, 1]. Non-zero values of $R_{fExt}(v)$ mean that v is near obstacles and therefore is not a free vertex.

Definition 6 (Succeeded connection ratio of a vertex). For a vertex, v, its succeeded connection ratio, $R_{sCon}(v)$, is the ratio of the number of succeeded connections from v and the total number of connections made from v, such as

$$R_{sCon}(v) = \frac{N_{sCon}(v)}{N_{fCon}(v)+N_{scon}(v)}, \tag{6}$$

where $N_{fCon}(v)$ is the number of failed connections from v and $N_{sCon}(v)$ stands for the number of succeeded connections from v. $R_{sCon}(v)$ takes values from [0, 1]. If the value of $R_{sCon}(v)$ is close to 1, it is very likely that region around v is free.

Definition 7 (Failed extension ratio of a collection of vertices). Given a vertex, *v*, let $S_d(v) = \{v_0, v_1, \cdots v_n\}$ be a set of vertices consisting of *v* and its neighbouring vertices within the circle centred at *v* and with a radius *d*. The failed extension ratio of $S_d(v)$, $R_{fExt}(S_d(v))$, is the ratio of the total number of failed extensions from every vertex in $S_d(v)$ and the total number of extensions made from $v_i, (i = 0, \cdots, n)$, such as

$$R_{fExt}(S_d(v)) = \frac{\sum_i N_{fExt}(v_i)}{\sum_i (N_{fExt}(v_i) + N_{sExt}(v_i))}, \tag{7}$$

where $\sum_i N_{fExt}(v_i)$ is the total number of failed extensions from $S_d(v)$, $\sum_i N_{sExt}(v_i)$ stands for the total number of succeeded extensions from $S_d(v)$, and *d* is the summation of the extension step and the minimal successful distance. $R_{fExt}(S_d(v))$ takes values from [0, 1]. The vertex *v* is a free vertex, if both $R_{fExt}(v)$ and $R_{fExt}(S_d(v))$ are zero.

Definition 8 (Succeeded connection ratio of a collection of vertices). Given a vertex, *v*, let $S_d(v) - \{v_0, v_1, \cdots v_n\}$ be a set of vertices consisting of *v* and its neighbouring vertices within the circle centred at *v* and with a radius *d*. The succeeded connection ratio of $S_d(v)$, $R_{sCon}(S_d(v))$, is the ratio of the total number of succeeded connections from every vertex in $S_d(v)$ and the total number of connections made from $v_i, (i = 0, \cdots, n)$, such as

$$R_{sCon}(S_d(v)) = \frac{\sum_i N_{sCon}(v_i)}{\sum_i (N_{fCon}(v_i) + N_{sCon}(v_i))}, \tag{8}$$

where $\sum_i N_{fCon}(v_i)$ is the total number of failed connections from $S_d(v)$ and $\sum_i N_{sCon}(v_i)$ stands for the total number of succeeded extensions from $S_d(v)$. $R_{sCon}(S_d(v))$ takes values from [0, 1]. The vertex v is likely to be a near-obstacle vertex if $R_{sCon}(S_d(v))$ is close to 1 and $R_{fExt}(v)$ takes a non-zero value.

Definition 9 (Free vertices ratio of a collection of vertices). Given a vertex, *v*, let $S_d(v) = \{v_0, v_1, \cdots v_n\}$ be a set of vertices consisting of *v* and its neighbouring vertices within the circle centred at *v* and with a radius *d*. The free vertices ratio of $S_d(v)$, $R_{free}(S_d(v))$, is the ratio of the number of free vertices and the total number of vertices within in $S_d(v)$, such as

$$R_{free}(S_d(v)) = \frac{N_{free}(S_d(v))}{N_{total}(S_d(v))}, \tag{9}$$

where $N_{free}(S_d(v))$ is the number of free vertices in $S_d(v)$ and $N_{total}(S_d(v))$ stands for the total number of vertices in $S_d(v)$. $R_{free}(S_d(v))$ takes values from [0, 1]. The vertex *v* is a near-obstacle vertex if $R_{free}(S_d(v))$ is not zero.

Definition 10 (Non-free vertices ratio of a collection of vertices). Given a vertex, *v*, let $S_d(v) = \{v_0, v_1, \cdots v_n\}$ be a set of vertices consisting of *v* and its neighbouring vertices within the circle centred at *v* and with a radius *d*. The non-free vertices ratio of $S_d(v)$, $R_{\overline{free}}(S_d(v))$, is the ratio of the number of non-free vertices and the total number of vertices within in $S_d(v)$, such as

$$R_{\overline{free}}(S_d(v)) = \frac{N_{\overline{free}}(S_d(v))}{N_{total}(S_d(v))}, \tag{10}$$

where $N_{\overline{free}}(S_d(v))$ is the number of non-free vertices in $S_d(v)$. $R_{\overline{free}}(S_d(v) = 1 - R_{free}(R_d(v))$.

Given a vertex and an extension step, the radius d can be calculated with the minimal successful connection distance returned by the RRT_connection function and subsequently $S_d(v)$ can be determined. With the obtained $S_d(v)$ and the returned information from both the RRT_extension and the RRT_connection functions, the values for all the features of the vertex can then be calculated.

3.3. CLASSIFICATION

A decision tree was developed to classify the types of the recorded vertices. The decision tree has the features as the nodes and the vertex types as the leaves.

A decision tree is normally developed through a training process with a training data set of a set of classes. Each datum in the dataset is labelled and consists of a set of features and a class of the class set. In the training process, information gain in terms of entropy is calculated for all the features and the one with the highest entropy is added to the tree as a new node.

As there exist a number of decision tree training algorithms, the key to the development of the decision tree for vertex type classification is the preparation of a training dataset. The class set, S_{class}, and the feature set, $S_{feature}$, to define the training date are shown in the following, respectively.

$$S_{class} = \{\text{free, near-obstacle, cluttered, narrow-passage}\},$$
$$S_{feature} = \{R_{fExt}(v), R_{fExt}(S_d(v)), R_{fExt}(S_d(v)), R_{sCon}(S_d(v)), R_{free}(S_d(v)), R_{\overline{free}}(S_d(v))\}.$$

Three simulation workspaces were set up for training data collection, namely, simple workspace, cluttered workspace and narrow-passage workspace, as shown in Figs. 9 to 11. In Fig. 9, the distance between every two obstacles is set at least

thee double of the extension step to ensure that all the vertices generated are either free vertices or near-obstacle vertices. In Fig. 10, a cluster of obstacles was set up and the distance between every pair of obstacles is shorter than the double of the extension step so that no free vertices can be generated inside the obstacle cluster. All the vertices inside the obstacle sets cluster will be cluttered vertices. Figure 11 shows a narrow passage going through an obstacle. Both the height and the width are smaller than the extension step. The length is greater than the minimal successful connection distance. Therefore, the passage is a narrow passage and all the vertices inside of the passage are narrow-passage vertices. To general training data, a simple incremental learning path planner which employs the RRT_extension function and the RRT_connection function was used to construct roadmaps in the given workspaces while answering path queries. To ensure the training data for each workspace are generic, all the queries were generated randomly. The path planner calculated the values of the features defined in Section 3.2 for all recorded vertices while constructing a roadmap until the vertices of the roadmap are distributed across the whole workspace. All recorded vertices were manually labelled with classes of S_{class}. The obtained values and the labels are then put together to form training data.

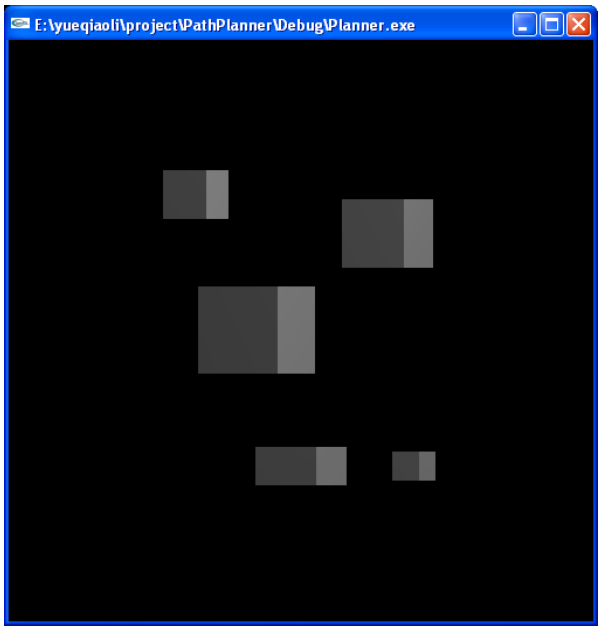

Fig. 9 – A simple workspace.

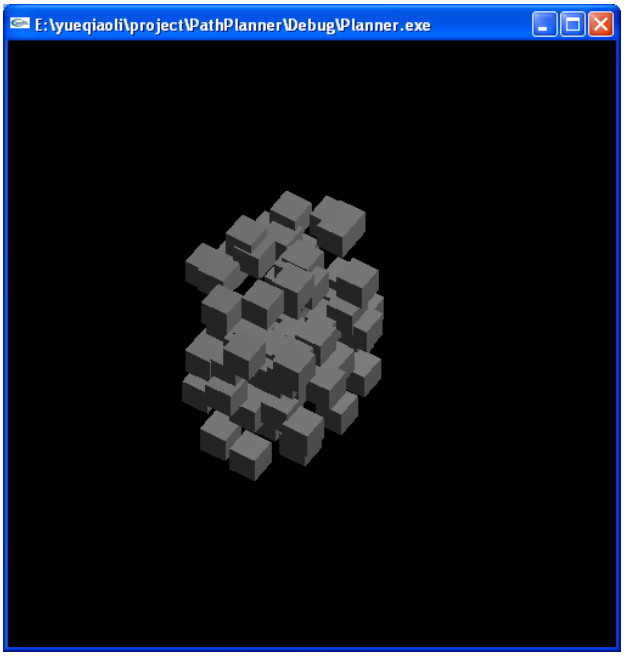

Fig. 10 – A cluttered workspace.

Fig. 11 – A narrow-passage workspace.

The training data were presented to C5.0, a well-developed decision tree training algorithm (Quinlan, 2007). The decision tree developed is shown in Fig. 12.

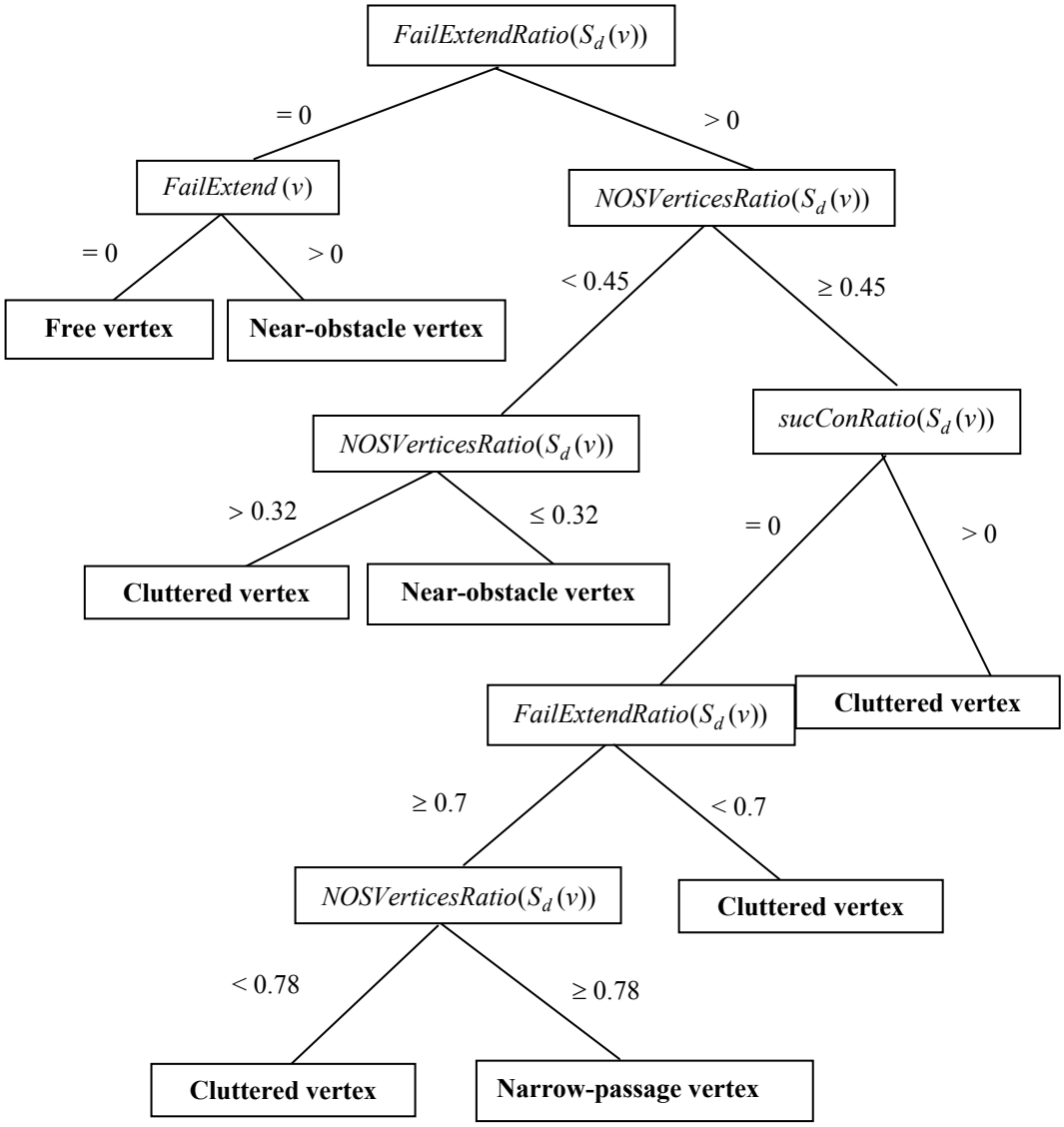

Fig. 12 – Vertex type classifier.

The decision tree was evaluated with another workspace, as shown in Fig. 13. The workspace contains free regions, cluttered regions and narrow passages. All different types of vertices could be generated in such a workspace.

Three groups of testing data were generated with different extension steps and successful connection distances using the same workspace. The value of the extension steps and the successful connection distances were set to make sure that all kind of vertices can be generated when the path planner constructs roadmaps.

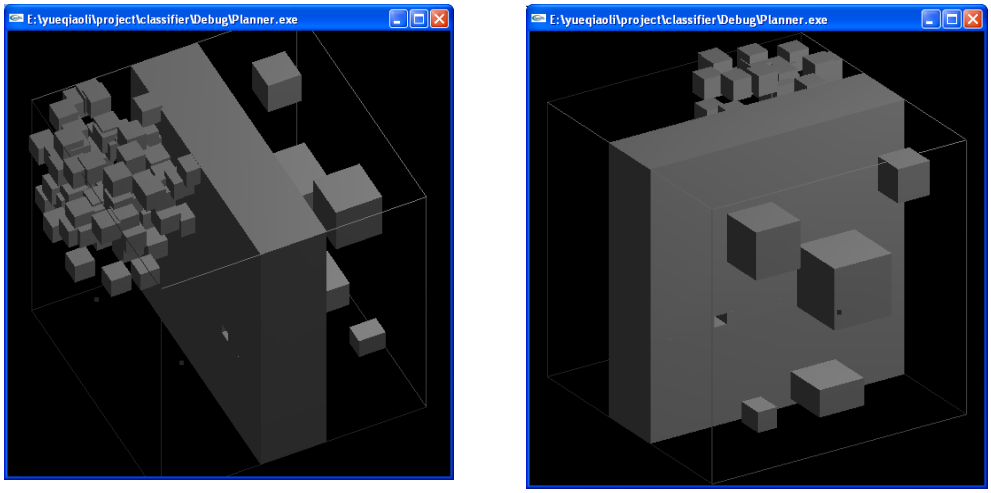

Fig. 13 – Evaluation experimental environment.

Simulation results are given in Table 1. The high success rates (≥ 80%) show that the vertex type classifier is able to classify the vertices in a roadmap in various regions. It also shows that the classifier works well with various extension steps and successful connection distances.

Table 1

Decision tree evaluation results

Robot size	Extension step	Successful connection distance	Successful rate
5*5*7	5	3* *extendStep* =15	88.1%
4.5*4.5*7	6	2.5* *extendStep* =15	89.0%
3.5*3.5*7	7	2* *extendStep* =14	87.4%

4. VERTEX PRUNING

Vertex pruning aims to produce high quality roadmaps. During the pruning process, the vertices that are useful to the construction of high quality roadmaps

will be retained and the rest will be removed. Incremental path planners allow roadmaps to be to be expended by having new vertices and edges added. While the feature values of some vertices of the existing roadmaps, obtained when the planners answered previous queries, will not be affected by the newly added vertices, others will because, for example, they are connected to the new vertices. Pruning only applies to the new vertices and those whose feature values changed when new vertices are added.

Vertices are removed according to their usefulness to a high quality roadmap. The usefulness depends on the quality measure and the vertex type which reflect the type of the region where the vertex resides. The new vertices and the existing ones whose feature values are affected by adding new vertices are grouped according to their types. The groups can be classified into free group, cluttered group, narrow-passage group and near-obstacle group. All the vertices in a group are connected with each other. Therefore, a group is a connected mini-roadmap. The vertices that connect to the vertices of other groups are marked as group connectors. Figure 14 shows all four groups; all the vertices in hollow points in each group are group connector vertices.

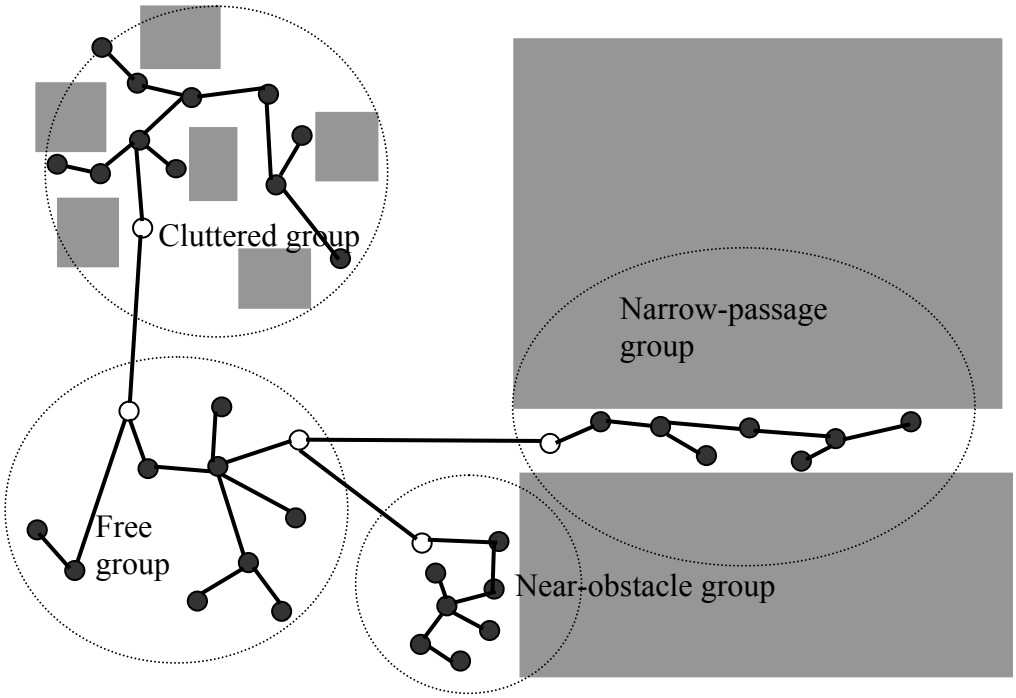

Fig. 14 – Vertices groups.

The quality criteria such as connectivity, roadmap size and useful circles are taken into the consideration of the usefulness for vertices in the free group. The following vertices are highly useful:

- Connectors – the connectors of a free group contribute of the connectivity of the group and the rest part of a roadmap as they link to the vertices of the rest part of the roadmap.
- Vertices that are in the paths from one connector to another – these vertices contribute to the connectivity within a free group. Without these vertices, paths from one connector to another cannot be formed.
- Vertices that form *K*-order deformation circles – they ensure useful circles are not broken.

CCR is not considered for the free group. This is because a free region does not contain any obstacle and is fully visible.

CCR, connectivity, roadmap size and useful circles are taken into the consideration of the usefulness for vertices in the cluttered group. The following vertices are highly useful:

- Connectors – the connectors of a cluttered group contribute of the connectivity of the group and the rest part of a roadmap as they link to the vertices of the rest part of the roadmap.
- Vertices that are in the paths from one connector to another – these vertices contribute to the connectivity within a cluttered group. Without these vertices, paths from one connector to another cannot be formed.
- Vertices that form *K*-order deformation circles – they ensure useful circles are not broken.

The following vertices are less useful:

- Sibling vertices of a leaf vertex within the range specified by the length of extension step – these vertices do not have added values to high CCR as they are too close to the leaf vertex.
- Leaf vertices if the distance between a leaf vertex and its ancestor is shorter than two times of the length of extension step – the leaf vertices is considered too close to their ancestors and do not make further contributions to CCR.

Connectivity and roadmap size are taken into the consideration of the usefulness for vertices in the narrow-passage group. The following vertices are highly useful:

- Connectors – the connectors of a narrow-passage group contribute of the connectivity of the group and the rest part of a roadmap as they link to the vertices of the rest part of the roadmap.
- Vertices that are in the paths from one connector to another – these vertices contribute to the connectivity within a narrow-passage group. Without these vertices, paths from one connector to another cannot be formed.

CCR, connectivity, roadmap size and useful circles are taken into the consideration of the usefulness for vertices in the near-obstacle group. The following vertices are highly useful:

- Connectors – the connectors of a near-obstacle group contribute of the connectivity of the group and the rest part of a roadmap as they link to the vertices of the rest part of the roadmap.
- Vertices that are in the paths from one connector to another – these vertices contribute to the connectivity within a near-obstacle group. Without these vertices, paths from one connector to another cannot be formed.
- Vertices that form *K*-order deformation circles – they ensure useful circles are not broken.
- Vertices whose N_{fExt} is not zero – these vertices are located along the edges of obstacles and contribute to CCR.

5. SIMULATION AND ANALYSIS

The simulation results represented in this section show the performance of the classifier when it works with a KSR incremental path planner (Li *et al*., 2008). The KSR path planner is able to produce high quality roadmaps by employing the vertex-based classifier to classify vertices of a roadmap and to remove the undesired vertices accordingly.

5.1. SCENARIOS

Simulation scenarios were set up for robots with high dof (≥ 3). The scenarios contain a variety of different workspaces regions types introduced in Section 2. Roadmaps that represent these regions consist of vertices which belong to various vertex types. The scenarios are as follows:

a) Simple scenario – the simple scenario represents a box robot with three dofs in a workspace consisting of nine cubical obstacles, which is illustrated in Fig. 15. The distance between each pair of obstacles is relatively large, compared to the size of the robot. Therefore, the roadmap contains free vertices and near-obstacle vertices.
b) Cluttered scenario – the cluttered scenario represents the same box robot moving in a workspace consisting of five hundred uniformly distributed tetrahedral obstacles, as shown in Fig. 16. The roadmap contains cluttered vertices only.

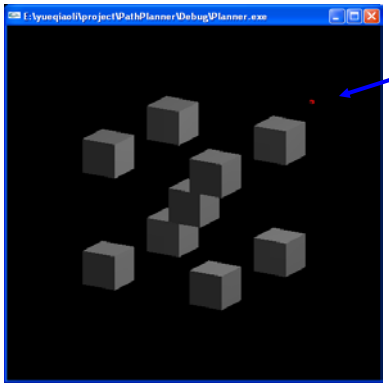

Fig. 15 – Simple scenario.

Fig. 16 – Clutter scenario.

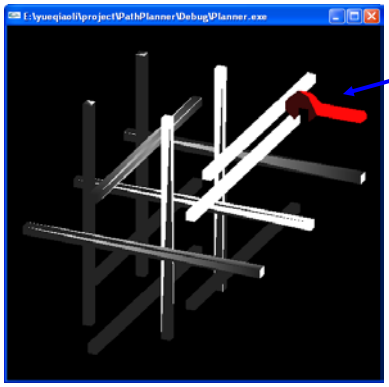

Fig. 17 – Wrench scenario.

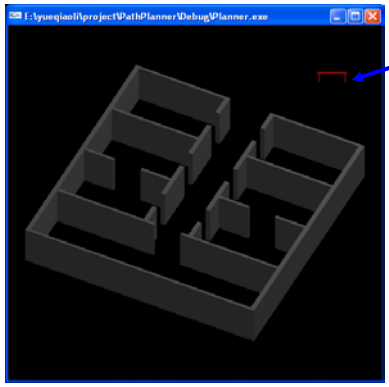

Fig. 18 – Room scenario.

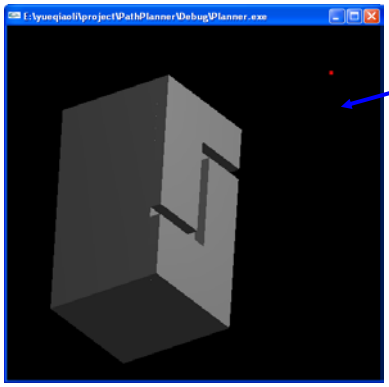

Fig. 19 – Tube scenario.

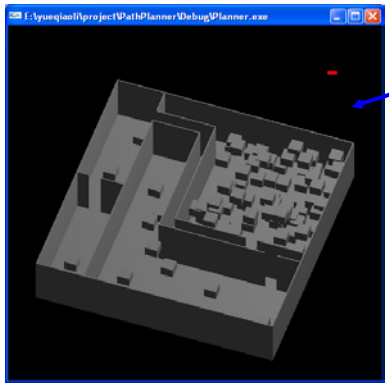

Fig. 20 – House scenario.

c) Wrench scenario: the wrench scenario represents a wrench robot with six dofs in a workspace consisting of twelve obstacles, as shown in Fig. 17. The size of the robot is relatively large compared to the spaces between the obstacles. The movements of the robot are constrained in such a workspace. Therefore, the roadmap contains cluttered vertices.
d) Room scenario – the room scenario represents a table robot with six dofs in an environment consisting of eight rooms, which is depicted in Fig. 18. The width of the doors and that of the corridor connecting the rooms are narrow compared to the size of the robot. The table robot is required to rotate in order to pass through the doors and to move along the corridor. However, since the doors are thin and the corridor between doors is short, the vertices in the roadmap are cluttered vertices. Therefore, the roadmap contains free vertices, cluttered vertices and near-obstacle vertices.
e) Tube scenario – the tube scenario represents a box robot with three dofs in a workspace consisting of a tube obstacle, which is shown in Fig. 19. The tube connects two obstacle free areas. There is a narrow passage inside the tube. This narrow passage is not straight which means that it is more difficult for the robot to travel through. The roadmap contains free vertices, narrow-passage vertices and near-obstacle vertices.
f) House scenario – the house scenario represents a bird robot with six dofs in a house environment as shown in Fig. 20. There are rooms and narrow corridors in the house. One of the rooms contains a number of pieces of furniture to construct a cluttered workspace. The roadmap contains all types of vertices.

5.2. CLASSIFICATION AND PRUNING

To evaluate the successful classification rate of the classifier when it works with the KSR planner, 100 path queries which are randomly generated were fed to the planner. Table 1 gives the successful classification rate achieved by the classifier together with the size of the roadmaps constructed by the KSR after 5, 10, 20, 50, 80, 100 queries were answered. To calculate the successful classification rate, all vertices were labelled with correct types manually in an off-line manner. The types of vertices identified by the classifier were then compared with the correct ones. The successful classification rate was calculated using the following formula:

$$R_s = \frac{N_s}{S_{roadmap}}. \tag{11}$$

The high success rate (above 80%) in Table 1 shows that the classifier can be used together with incremental path planners. The successful classification rate increases along with more path queries are answered. As the KSR is an incremental

path planner, region information stored in a vertex was insufficient at the beginning to reflect the workspace surrounding the vertex, which leads to the incorrect classification.

Table 2

Success classification rate of the vertex category classifier

		Workspace					
Query number	KSR Size & Success rate	Simple	Cluttered	Wrench	Room	Tube	House
5	size	34	119	145	134	160	224
	Success rate	90.0.3%	80.7%	82.8%	81.5%	80.4%	76.1%
10	size	87	299	369	236	239	364
	Success rate	95.5%	82.3%	83.2%	84.9%	88.1%	78.1%
20	size	135	661	711	375	276	514
	Success rate	96.1%	83.5%	84.4%	85.3%	88.7%	80.4%
50	size	261	997	883	411	296	632
	Success rate	96.6%	84.1%	85.1%	86.5%	88.7%	83.1%
80	size	261	1241	956	462	305	710
	Success rate	96.6%	85.8%	85.1%	86.5%	88.7%	83.9%
100	size	261	1304	992	487	305	713
	Success rate	96.6%	85.8%	85.1%	86.5%	98.7%	83.9%

The classifier updated the classification results and corrected the incorrect classification with more region information obtained when more vertices are added by the KSR path planner.

Pruning based on the classification significantly reduced the size of roadmaps constructed by the planner and at the same time retained the quality of the roadmaps. Table 3 shows the comparison on the quality of the roadmaps before and after the pruning process.

The quality of roadmaps built up based on the classification and pruning was measured against CCR, connectivity, useful circles and roadmap size. The performance of the KSR path planner, in teams of the efficiency of answering queries by using the roadmaps was measured with the number of collision detection carried out by the KSR path planner.

Since the KSR path planner is an incremental path planning algorithm, the quality of the roadmaps improves markedly during the first few path queries. It then changes slightly when the roadmaps are able to answer most of the path queries. Therefore, the measures of roadmap quality were recorded after 5, 10, 20, 50, 80, 100 path queries were answered in each scenario.

Table 3

The performance of the KSR path planner with and without pruning

Workspace	Pruning	size	Connectivity	CCR	Useful cycles
Simple	no	772	1	100%	13
	yes	261	1	100%	12
Clutter	no	2929	1	100%	47
	yes	1304	1	100%	43
Wrench	no	2528	1	100%	17
	yes	992	1	100%	16
Room	no	1652	1	100%	3
	yes	487	1	100%	3
Tube	no	2532	1	100%	0
	yes	305	1	100%	0
House	no	1783	1	100%	25
	yes	713	1	100%	22

5.3. QUALITY OF ROADMAPS

The number of collision detection decreased along with more and more queries were answered. This is because CCR increased when more queries were answered. The number of collision detection made by the KSR planner was also recorded after 5, 10, 20, 50, 80, 100 path queries were answered in each scenario. Tables 4 to 9 show the quality of the roadmaps and the performance of the KSR planner using the roadmaps.

Table 4

The performance of the KSR path planner in the simple scenario

Quality	Num of query	Size	Connectivity	Coverage	Useful cycles	Collision detections/query
Simple	5	34	1	93.2%	3	76
	10	87	1	98.3%	9	36
	20	135	1	98.3%	10	40
	50	261	1	100.0%	12	24
	80	261	1	100.0%	12	12
	100	261	1	100.0%	12	10

Table 5

The performance of the KSR path planner in the clutter scenario

Quality	Num of query	Size	Connectivity	Coverage	Useful cycles	Collision detections/query
Clutter	5	119	3	45.5%	1	288
	10	299	2	63.2%	5	245

	20	661	1	82.8%	16	192
	50	997	1	95.6%	35	131
	80	1241	1	100.0%	39	81
	100	1304	1	100.0%	43	11

Table 6

The performance of the KSR path planner in the wrench scenario

Quality	Num of query	Size	Connectivity	Coverage	Useful cycles	Collision detections/query
Wrench	5	145	1	73.2%	0	210
	10	369	1	89.4%	2	187
	20	711	2	96.1%	4	167
	50	883	1	100.0%	6	98
	80	956	1	100.0%	13	54
	100	992	1	100.0%	16	14

Table 7

The performance of the KSR path planner in the room scenario

Quality	Num of query	Size	Connectivity	Coverage	Useful cycles	Collision detections/query
Room	5	134	4	43.9%	0	183
	10	236	2	74.8%	0	164
	20	375	2	87.3%	2	150
	50	411	1	100.0%	2	98
	80	462	1	100.0%	3	45
	100	487	1	100.0%	3	13

Table 8

The performance of the KSR path planner in the tube scenario

Quality	Num of query	Size	Connectivity	Coverage	Useful cycles	Collision detections/query
Tube	5	160	2	56.7%	0	185
	10	239	2	91.2%	0	136
	20	376	1	100.0%	0	111
	50	296	1	100.0%	0	45
	80	305	1	100.0%	0	31
	100	305	1	100.0%	0	24

Table 9

The performance of the KSR path planner in the house scenario

Quality	Num of query	Size	Connectivity	Coverage	Useful cycles	Collision detections/query
House	5	224	3	36.7%	0	265
	10	364	2	56.8%	3	210
	20	514	1	77.6%	6	138
	50	632	1	92.3%	14	81
	80	710	1	100.0%	20	27
	100	713	1	100.0%	22	16

6. CONCLUSION

This paper emphasises the construction of high quality roadmaps. Incremental path planners are able to build up high quality roadmaps if vertices that are not useful to the quality can be removed during the process of roadmap development. The usefulness of vertices depends on the types of regions of a workspace where the vertices locate. The presented vertex-based classifier is able to classify the types of vertices, which indicate the types of regions, to high successful rates. Integrating the classifier into a KSR incremental path planner leads to high quality roadmaps in various types of workspaces. Simulations show that the quality of roadmaps was gradually improved during the incremental process of roadmap construction.

REFERENCES

1. Adorno, B.V. and Borges, G.A., *iARW: An incremental path planner algorithm based on adaptive random walks*, Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, MO, USA, Issue 2, pp. 988-993.
2. Dale, L.K. and Amato, N. M., *Probabilistic roadmaps-putting it all together*, Proceedings of The 2001 IEEE International Conference on Robotics and Automation, 21-26 May, 2001, Seoul, Korea, Vol. 2, pp. 1940-1947.
3. Denny, J. Greco, E. Tapia, L. and Amato, N. M., *Region identification methods for efficient and automated motion planning*, Technical Report TR10-002, Parasol Lab., Department of Computer Science and Engineering, Texas A&M University, 2009.
4. Geraerts, R., *Sampling-based Motion Planning: Analysis and Path Quality*, PhD Thesis, Utrecht University, 2006.
5. Jaillet, L. and Siméon, T., *Path Deformation Roadmaps: Compact Graphs with Useful Cycles for Motion Planning*, International Journal of Robotics Research, **27**, *11-12*, pp. 1175-1188, 2008.
6. Kazemi, M., Mehrandezh, M, and Gupta, K., *An Incremental Harmonic Function-based Probabilistic Roadmap Approach to Robot Path Planning*, Proceedings of The 2005 IEEE

International Conference on Robotics and Automation, April 18-22, 2005, Barcelona, Spain, pp. 2136-2141.

7. Kuffner, J.J. and LaValle, S.M., *RRT-connect: an efficient approach to single-query path planning*, Proceedings of The 2000 IEEE International Conference on Robotics and Automation, April 24-28, 2000, San Francisco, California, USA, *2*, pp. 995-1001, 2000.
8. LaValle, S.M., *Rapidly-exploring random trees: a new tool for path planning*, Technical Report 98-11, Computer Science Department, Iowa State University, October 1998.
9. Li, Y. Li, D, Maple, C. and Yong Y., *K-order Surrounding Roadmaps for Robot Path Planning*, Proceedings of Human Adaptive Mechatronics Workshop, January, 2009, Stafford, UK, 2009.
10. Li, T.Y. and Shie, Y.C., *An incremental learning approach to motion planning with roadmap management*, The Proceedings of the IEEE International Conference on Robotics and Automation, May 11-15, 2002, Washington DC, USA, **4**, pp. 4311-3416.
11. Morales, A., Tapia, L., Pearce, R., Rodriguez, S., & Amato, N.M., *C-space subdivision and integration in feature-sensitive motion planning*, Proceedings of The 2005 IEEE International Conference on Robotics and Automation, April 18-22, 2005, Barcelona, Spain, pp. 3114-3119.
12. Nieuwenhuisen, D. and Overmars, M.H., *Useful cycles in probabilistic roadmap graphs*, Proceedings of the 2004 IEEE International Conference on Robotics & Automation, April 16-May 1, 2004, New Orleans, LA, USA, pp. 446-452.
13. Siméon, T., Laumond, J. P. and Nissoux, C., *Visibility based probabilistic roadmaps for motion planning*, Advanced Robotics Journal, **14**, *6*, pp. 477-493, 2000.
14. Schmitzberger, E., Bouchet, J. L., Dufaut, M., Wolf, D. and Husson, R., *Capture of homotopy classes with probabilistic roadmap*, Proceedings of the 2002 IEEE International Conference on Intelligent Robots and System (May 11-15, 2002), Washington DC, USA, **3**, pp. 2317-2322, 2002.