https://doi.org/10.31436/iiumej.v24i1.2529

AN INTEGRATED RRT*SMART-A* ALGORITHM FOR SOLVING THE GLOBAL PATH PLANNING PROBLEM IN A STATIC ENVIRONMENT

HERU SUWOYO[1], ANDI ADRIANSYAH[1 *], JULPRI ANDIKA[1], ABU UBAIDAH SHAMSUDIN[2] AND MOHAMAD FAUZI ZAKARIA[2]

[1]Department of Electrical Engineering, Universitas Mercu Buana, Jakarta, Indonesia
[2]Department of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia

[*]Corresponding author: andi@mercubuana.ac.id

(Received: 15th July 2021; Accepted: 5th October 2022; Published on-line: 4th January 2023)

ABSTRACT: The use of sampling-based algorithms such as Rapidly-Exploring Random Tree Star (RRT*) has been widely applied in robot path planning. Although this variant of RRT offers asymptotic optimality, its use is increasingly limited because it suffers from convergence rates, mainly when applied to an environment with a poor level of obstacle neatness and a narrow area to the target. Thus, RRT*-Smart, a further development of RRT*, is considered ideal for solving RRT* problems. Unlike RRT*, RRT*-Smart applies a path optimization by removing the redundant nodes from the initial path when it is gained. Moreover, the path is also improved by identifying the beacon nodes used to steer the bias of intelligent sampling. Nevertheless, this initial path is found with termination criteria in terms of a region around the goal node. Consequently, it risks failing to generate a path on a narrow channel. Therefore, a novel algorithm achieved by combining RRT*-Smart and A* is proposed. This combination is intended to switch method-by-method for the exploration process when the new node reaches the region around the goal node. However, before RRT*-Smart is combined with A*, it is improved by replacing the random sampling method with Fast Sampling. In short, by involving A*, the exploration process for generating the Smart-RRT*'s initial path can be supported. It gives the optimal and feasible raw solution for any complex environment. It is logically realistic because A* searches and evaluates all neighbors of a current node when finding the node with low cost to the start and goal node for each iteration. Therefore, the risk of collision with an obstacle in the goal region is covered, and generating an initial path in the narrow channel can be handled. Furthermore, this proposed method's optimality and fast convergence rate are satisfied.

ABSTRAK: Penggunaan algoritma berasaskan pensampelan seperti Rapidly-Exploring Random Tree Star (RRT*) telah digunakan secara meluas dalam perancangan laluan robot. Walaupun varian RRT ini menawarkan keoptimuman tanpa gejala, penggunaannya semakin terhad kerana ia mengalami kadar penumpuan, terutamanya apabila digunakan pada persekitaran dengan tahap kekemasan halangan yang lemah dan kawasan yang sempit ke sasaran. Oleh itu, RRT*-Smart, pembangunan lanjut RRT*, dianggap sesuai untuk menyelesaikan masalah RRT*. Tidak seperti RRT*, RRT*-Smart menggunakan pengoptimuman laluan dengan mengalih keluar nod berlebihan daripada laluan awal apabila ia diperoleh. Selain itu, laluan juga dipertingkatkan dengan mengenal pasti nod suar yang digunakan untuk mengemudi bias pensampelan pintar. Namun begitu, laluan awal ini ditemui dengan kriteria penamatan dari segi rantau di sekeliling nod matlamat. Akibatnya, ia berisiko gagal menjana laluan pada saluran yang sempit. Oleh itu, algoritma baru yang dicapai dengan menggabungkan RRT*-Smart dan A*

dicadangkan. Gabungan ini bertujuan untuk menukar kaedah demi kaedah untuk proses penerokaan apabila nod baharu sampai ke kawasan sekitar nod matlamat. Walau bagaimanapun, sebelum RRT*-Smart digabungkan dengan A*, ia diperbaiki dengan menggantikan kaedah persampelan rawak dengan Persampelan Pantas. Pendek kata, dengan melibatkan A*, proses penerokaan dalam menjana laluan awal yang Smart-RRT lakukan* boleh disokong. Ia memberikan penyelesaian mentah yang optimum dan boleh dilaksanakan untuk mana-mana persekitaran yang kompleks. Ia adalah realistik secara logik kerana A* mencari dan menilai semua jiran nod semasa apabila mencari nod dengan kos rendah ke nod permulaan dan matlamat untuk setiap lelaran. Oleh itu, risiko perlanggaran dengan halangan di kawasan matlamat dilindungi, dan menjana laluan awal dalam saluran sempit boleh dikendalikan. Tambahan pula, kaedah optimum yang dicadangkan dan kadar penumpuan yang cepat ini berpuas hati.

KEYWORDS: *path planning; A* algorithm; RRT*-smart algorithm; convergence rate*

1. INTRODUCTION

As an intelligent application, mobile robots have received considerable attention from researchers. The increase in productivity and efficiency offered by the use of mobile robots is behind this concern [1-3]. Therefore, it is not surprising that developments in navigation on mobile robots continue rapidly because navigation is at the core of its intelligence [4,5]. Path planning is the basis of a robotic navigation, namely the ability to plan a path without collisions with obstacles in the environment [6-8]. Although there are several strategies with different exploration representations, grid-based algorithms [9-11] and sampling-based algorithms [12-15] are in great demand because of their ease of implementation. For example, there are A* and RRT* algorithms for these types, respectively. As its advantage, A* offers the completeness and optimality of the resolution which is not guaranteed by RRT*. However, A* algorithm is limited in use, because it consumes much time to solve the problem when applied to high-dimensional state space. Accordingly, RRT* has been chosen to tackle this limitation. Although RRT* in this case is obviously better than A* algorithm, it works infinitely when the region around the goal node is not reached. It is also work based on the probabilistic principle leading the large variance to time which gives its another disadvantage named slow convergence rate. Moreover, RRT* uses the criteria termination in the form of a predefined maximum number of the sampling node and condition when the new explored node reaches the region around the goal node, which increases the processing time.

Referring to this limitation, RRT* have been intensively maintained [16]. This concern not only enhances the optimality of RRT* but also accelerates its convergence rate. There are several algorithms intended to tackle this limitation such as Fast-RRT [17], Connect-RRT [11,15,18], informed RRT*[11], and Smart-RRT*[16,17,19,20]. The most popular algorithm is Smart-RRT* with claims optimality and fast working time. Although it also applies some procedures of RRT*, the Smart-RRT* also executes the path optimization when the initial path is found [21]. This makes RRT*-smart a better solution compared to RRT* in terms of optimality. This optimization is designed to remove the redundant node given by the initial path. Moreover, RRT*-Smart is also completed with an intelligent sampling to replace the conventional random sampling [16]. Conceptually, this sampling is done by biasing it to the beacon node of optimized path, in which the beacon node is identified from the path optimization process. Therefore, once the RRT*-Smart obtains a shorter path compared to the previous one, the optimization is conducted and simultaneously gets the new beacon node. Accordingly, it improves its predecessor named RRT*.

https://doi.org/10.31436/iiumej.v24i1.2529

However, due to following some procedures of RRT*, RRT*-Smart takes over the particular characteristic of RRT* namely, the criteria termination [22]. Similar to RRT*, the process of getting the initial path is noticed to break when the newest node is in the region around the goal node. Therefore, both RRT* and RRT*-Smart are still at risk of failure in generating a feasible path when the start and goal node are separated with a narrow channel in the environment [23-26]. Furthermore, terminating the exploration process of Smart-RRT* with a region around a goal node requires the knowledge of any obstacle close to the goal node that would be difficult for the robot to perceive. The reason for this difficulty is because the characteristic of map information that might be given by Simultaneous Localization and Mapping is still categorized as raw. Thus, terminating the process with a trigger of radius will definitely cause RRT*-Smart to make a potential mistake.

For this urgency, it is strongly recommended to improve RRT*-Smart with an ability of connecting the newly expanded node to the goal node when they are closed to each other in the channel environment. Through this paper, aiming to tackle the limitation in terms of optimality and keep the way of determining the optimized path, A* algorithm is proposed to handle the rest task of path planning when the newly expanded node reaches the area closed to the goal node even when a border exists. A* algorithm has been recorded as a searching-based planning method with some advantages such as having optimality and high resolution of tracing procedure [27]. This advantage makes the A* algorithm relevant to be applied as a helper for RRT*-Smart in narrow channels near the goal node so that the time required for search expansion can be further reduced. In general, the search process using this combination begins by using RRT*, part of the RRT*-Smart process, to carry out the exploration stage. In contrast to RRT* and RRT*-Smart, this case uses a fast-sampling technique to replace conventional sampling technique. This is intended to speed up the exploration process and achieve node positions near the goal. For the record, fast sampling is a technique applied to fast-RRT, which performs the exploration process without repeating the placement of sampling nodes in areas close to a group of nodes that have been recorded. While this represents an advantage for searching in a wide environment, the potential difficulty in expanding into narrow channels increases. For this reason, the expansion process needs to be supported by the relevant algorithm, namely A*. The formation of this initial path will be completed after A* makes a connection from the closest node to the goal with the goal node. Furthermore, the proposed algorithm will continue the process by applying path optimization and intelligent sampling from RRT*-Smart, so that optimality is maintained with an increased convergence rate.

2. MATERIAL AND METHODS

As usual, the global problem is defined to show the relation to its existing solution. It is reliable by firstly defining the objective of finding a solution for a path planning problem, which is how to determine the connection from the initial node to the goal node without any collision with any obstacle existing in the environment. Globally, there are two important benchmarks used for evaluating the performance of a path planning algorithm, namely optimality and convergence rate. Respectively, these can be represented by a feasible path and time consumption in generating it. Therefore, by well knowing the problem and these benchmarks, a feasibility of the designed or targeted algorithm could be declared.

https://doi.org/10.31436/iiumej.v24i1.2529

Let $Z \in \mathbb{R}^n$ be the representation of state space for a path planning problem, with $n \in N$ being the space dimension, thus $Z = \{Z_{obs,}Z_{free}\}$ is another state space with $Z_{obs} \in Z$ referring to obstacle coordinates and $Z_{free} \in X$ free space. Moreover, if the start node $z_{init} \in Z_{free}$ and goal node $z_{goal} \in Z_{free}$ are given, then referring to Z_{obs}, the path planning algorithm has to find the ideal path to/from those nodes, denoted as $\sigma = [0, T] \rightarrow Z_{free}$with $\sigma(0) = x_{init}$ and $\sigma(T) = Z_{goal}$ where $Z_{goal} = \{z \in Z \| z - z_{goal} \| < r\}$ where r is the radius around goal defined at the beginning.

2.1 A* Algorithm

A* is considered to be a heuristic-based search algorithm with the advantage of optimal path [28], [29]. This algorithm has a main step to assess the direction of travel by referring to the lowest cost/distance value from the neighbor *n*-th node around the current node $z_{current}$. This cost is obtained by scanning the distance from the current node with eight surrounding nodes to get the value of (n), and also from a heuristic function $h(n)$ that gives the value of h, which is the distance value from the eight nodes around the current node to the destination node, where this distance is calculated by applying Euclidean Distance and $f(n)$ is mathematically determined using Eq. (1).

$$f(n) = g(n) + h(n, z_{goal}) \tag{1}$$

where $g(n)$ is a function that represents the cost of the path required from the start node z_init to the current node $z\_current$. Briefly, the calculation of $g(n)$ refers to the eight costs which are determined based on the position in relation to the current that is the center. While $h(n, z_{goal})$ is a function used to calculate the required cost from the goal node z_{goal} to the n-*th* node. This function applies Euclidean distance which directly calculates the distance to the goal node regardless of whether there is a collision with an obstacle or not. The sum of both is called f_{cost} which is used as the evaluation function of the nth node. In general, there are two sets named $openSet$ and $closedSet$ in A*. All nodes that will be evaluated are placed in $openSet$, and then the nodes that have been evaluated are removed from $openSet$ and moved to $closedSet$. At the start of the process, the start node z_init is placed as a member of the $openSet$ set since the start node is the initial of the search then $g(z\_init) = 0$. Thus f_{cost} for the start node $f(z_{init})$ is the same as $h(z_{init}, z_{goal})$. Furthermore, taking into account all the evaluation costs of f_{cost} for all nodes in the $openSet$, the node with the lowest f_{cost} will be selected as the current node $z_{current}$ and its availability in the $openSet$ is removed and transferred/added to the $closedSet$. As a step in defining the termination criteria, if this current node is a goal node, the process is stopped because it explains that the path has been obtained. Instead, all neighbors of the current node will be checked based on their g_{cost}. In the case of the grid map, there are the eight-neighbor nodes for the current node. Each neighbor node has a specified cost to the current node and is used as the basis for calculating g_{cost}. If the neighbor node is in a $closedSet$ or it is not traversable, the scan is continued on the next neighbor node. The neighbor node with the lowest g_{cost} is determined. The neighbor node with the lowest g_{cost} will then be checked, if it is not in the $openSet$ then it is moved, set its f_{cost}, and set its parent to current node. Furthermore, this series of processes will be repeated until the termination criteria are met and to clarify this description the pseudocode of the A* algorithm is given

https://doi.org/10.31436/iiumej.v24i1.2529

Algorithm* 1. A^* *Algorithm

$ClosedSet = [\ \ ]$
$OpenSet = [\ \ ]$
$OpenSet = [z_{init}]$
$g(z_{init}) = 0$
$h(z_{init}) \leftarrow hcalculate(z_{init}, z_{goal})$
$f(z_{init}) \leftarrow 0 + h(z_{init})$
while $openSet \neq \emptyset$ **then**
  $z_{current} = node\ in\ OpenSet\ lowest\ f_{cost}$
  $remove\ z_{current}\ from\ OpenSet$
  $add\ z_{current}\ to\ ClosedSet$
  if $z_{current} = z_{goal}$

    $return$
  endif
  foreach $z_{neighbour}\ of\ z_{current}$
    if $z_{neighbour}\ is\ in\ ClosedSet$ **or** $z_{neighbour}\ is\ not\ traversable$
      $continue$
    endif
    if $newpath\ to\ z_{neighbour}\ is\ shorter$ **or** $z_{neighbour}\ is\ not\ in\ OpenSet$
      $set\ f_{cost}\ of\ neighbour$
      $set\ parent\ of\ neighbour\ to\ z_{current}$
      if $neighbour\ is\ not\ in\ OpenSet$
        $add\ neighbour\ to\ OpenSet$
      endif
    endif
  endforeach
end

2.2 RRT*-Smart

RRT*-Smart is an enhanced version of RRT* that works in the same way as RRT* for finding the initial path [14,20]. Besides that, it undertakes a path optimization procedure once an initial path has been determined. This procedure eliminates unnecessary nodes from the path that was initially discovered. The optimization is supported by the beacon nodes which are determined after conducting a triangular inequality technique when the initial path is found. Intelligent sampling is the second key feature introduced by RRT*- Smart. This sampling differs from random sampling in that it is directed towards optimal path beacon nodes. It sets a radius for intelligent exploration around selected beacons using a Biasing Radius b. RRT*-Smart performs the path optimization procedure again to build additional beacon nodes as soon as it identifies a shorter path. As a result, RRT*-Smart improves path cost and accelerates path convergence.

https://doi.org/10.31436/iiumej.v24i1.2529

Algorithm 2. RRT*Smart Algorithm $\sim$ $T = (V, E) \leftarrow RRT^*Smart(z_{init})$

1 $T \leftarrow InitializeTree(\,)$
2 . $T \leftarrow InsertNode(\emptyset, z_{init}, T)$
3 . **for** $i = 1$ **to** N **do**
4 **if** $i = n + b, n + 2b, n + 3b \ldots$ **then**
5 $z_{rand} \leftarrow intelligent\_sampling(i, z_{beacon})$
6 **else**
7 $z_{rand} \leftarrow sampling(i)$
8 **endif**
9 $z_{nearest} \leftarrow nearest(T, z_{rand})$
10 $(z_{new}, u_{new}, T) \leftarrow steer(z_{nearest}, z_{rand})$
11 **if** $obstaclefree(z_{new})$ **then**
12 $z_{near} \leftarrow near(T, z_{new}, |V|)$
13 $z_{min} \leftarrow choosenparent(z_{near}, z_{nearest}, z_{new})$
14 $T \leftarrow insertnode(z_{min}, z_{new}, T)$
15 $T \leftarrow rewire(T, Z_{near}, Z_{min}, Z_{new})$
16 **endif**
17 **if** $initialPathfound$ **then**
18 $n \leftarrow i$
19 $(T, directcost) \leftarrow pathOptimization(T, z_{init}, z_{goal})$
20 **endif**
21 **if** $(directcostnew < directcostold)$ **then**
22 $Z_{beacons} \leftarrow pathOptimization(T, z_{init}, z_{goal})$
23 **endif**
24 **return** T
25 **endfor**
26

In the initial path-finding stage, RRT*-Smart applies a conventional sampling technique. This technique will randomly generate nodes in the search space referring to the maximum and minimum limits of their representation. Based on these random nodes, the nearest node is determined by applying a Euclidean distance evaluation to the set of nodes. The direction from the nearest node to a random node is then determined as a reference direction for generating new nodes. In addition to referring to the reference direction, the placement of new node locations is also carried out based on the calculated distance between the random node and the nearest node. If the distance is less than or equal to the specified reference distance, then the placement of the new node from the nearest node is as far as the calculated distance. On the other hand, the placement of the new node from the nearest node is equal to the specified reference distance. This process is represented by Algorithm 2 in lines 9 and 10 after the sampling process in line 7 is carried out. Then RRT*-Smart will continue the process by paying attention to the new node. If the location of the new node is not the same as one of the points of the obstacle position and also the line formed from the new node to the nearest node does not intersect with any line from the obstacle, then the new node is connected to the nearest node as an Edge E. This process is called the wiring process in RRT. Furthermore, the nearest node will be temporarily considered as the parent node z_{near} of the new node z_{new} and all neighbor nodes of the new node are then evaluated for their proximity. The node that has the closest distance will be called z_{min} which is also the parent node for the new node. And the new node is linked to z_{min}. This stage is rewiring which simultaneously distinguishes RRT and RRT*, where the process is shown in lines 11 to 15 of Algorithm

2. In RRT*-Smart this series of steps is a process in determining the initial path before this path is optimized by applying Triangular Inequality. In short, this technique builds a new edge by connecting the node to the next node in the node set T if the relationship between the two nodes is free of obstacles and collisions. So, it is clear that in this process, the number of nodes will be reduced. This new set of nodes is called beacons $z_{beacons}$ as lies on line 23 and simultaneously the path formed from start to goal is connecting all these beacons with its costs termed $directcost$. Finding this initial path will trigger intelligent sampling with a number of samples spawned around $z_{beacons}$. This is done as an effort to reduce path costs by optimizing new nodes that are potentially better than the beacon position without having to do time-consuming sampling. Finally, the path correction is conducted in the global looping once the lower $directcost$ is found.

3. PROPOSED METHOD – RRT*SMART-A* ALGORITHM

In a high-dimensional space, as a sampling-based algorithm, RRT is indeed better and faster in finding the initial path. However, by observing more deeply, precisely by taking random samples, the variance of the search time is so large that it has the potential to take a long time to determine a feasible path. This is even more so in the narrow channel scenario, which of course will take a lot of time, because each path is formed randomly due to random sampling. It is this basis upon which RRT* is introduced. As previously mentioned, the difference between the two is that there are two operations on the RRT*, namely neighbors search and rewiring step after wiring step is done. The parent, which is always updated every time a new node is generated, certainly makes RRT* have the advantage of maintaining the optimality of the path. RRT*-Smart applies this to initial path determination. However, this kind of determination requires a lot of time and memory usage to achieve a truly optimal path. Accordingly, a new method with the name An Integrated RRT*Smart-A* Algorithm is introduced in this paper.

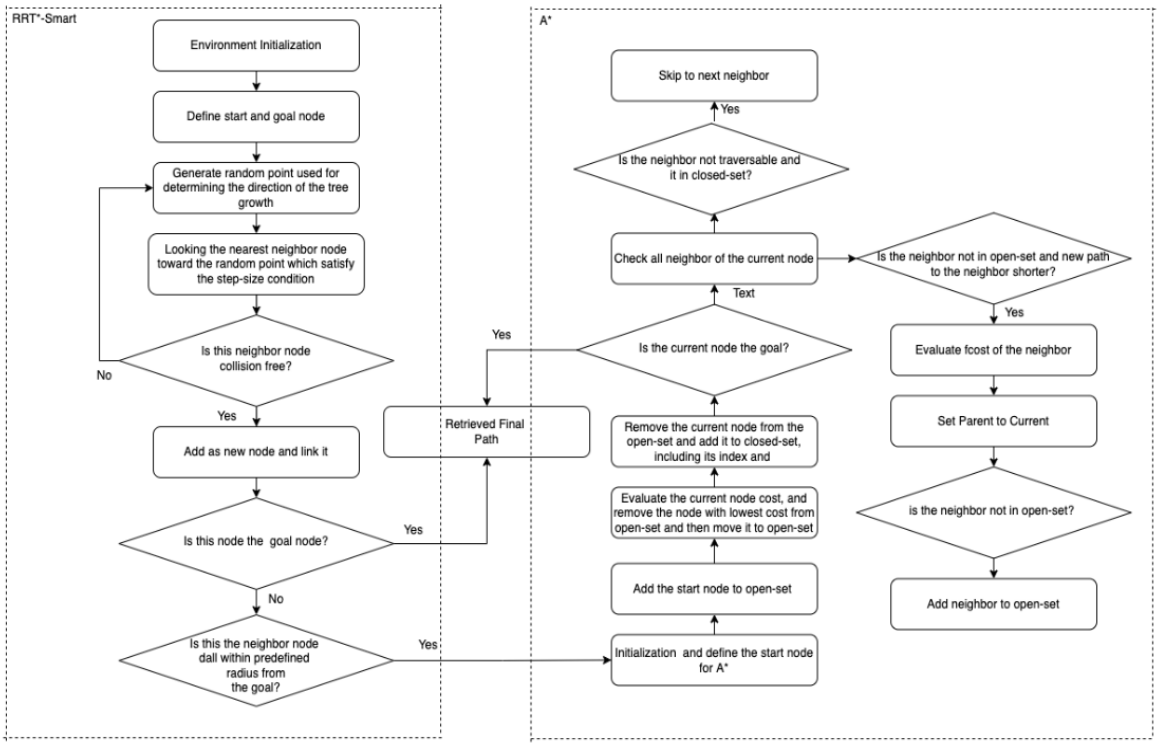

Fig. 1: Flowchart of Proposed Method (An Integrated RRT*-Smart A* Algorithm)

https://doi.org/10.31436/iiumej.v24i1.2529

As seen in Fig. 1, in addition to integrating two different approaches referring to the proximity of the last exploration node to the goal node, conventional sampling techniques in the early stages of RRT*-Smart were replaced with fast-sampling techniques. Instead of expanding on a new exploration area, random sampling is not limited to the possibility of falling on the explored area. This reason underlies the replacement of conventional sampling with fast sampling. Where fast-sampling applies restrictions in generating samples, random samples obtained will be rejected if their position is in the explored area, and only random samples are in the new exploration area (see Fig. 2 and Algorithm 3 lines 7 up to 10).

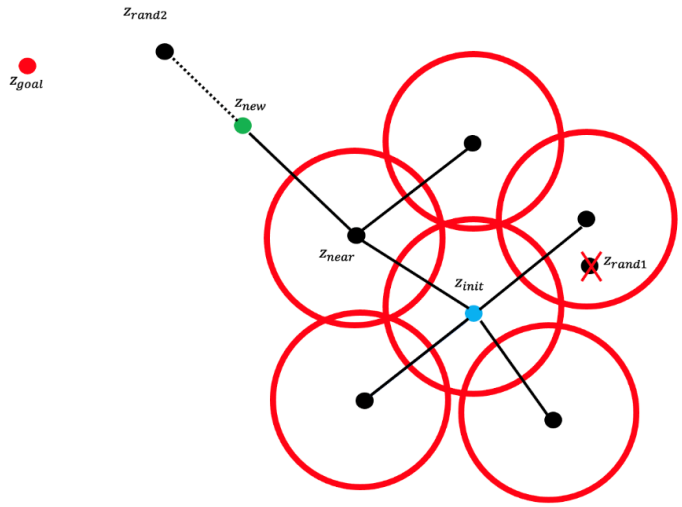

Fig. 2: Illustration of Fast-Sampling Technique. The red circle is the explored area. When random node z_{rand1} falls on the explored area it is rejected and random sampling is repeated. When the random node z_{rand2} is obtained, the wiring and rewiring process is carried out.

Furthermore, new nodes z_{new} formed during expansion will continue to be observed. If it is close to the goal node z_{node}, A* will take over determining this initial path. This aims to minimize the consumption of expansion time in narrow channels (see Algorithm 4 from lines 21 to 24). Furthermore, when the initial path is obtained, all feasible nodes will be directly connected by applying triangular equality. This application is intended to shorten the path as an optimization step. Iteration in this process starts from z_{goal} and moves to z_{init} by observing the direct connection to the parent sequentially until the connection of two different nodes is declared a collision on the obstacle. When the order of observations reaches z_{init} then no more nodes can be connected directly. To provide clarity on this process Fig. 3 and Algorithm 4 are given (see Algorithm 3).

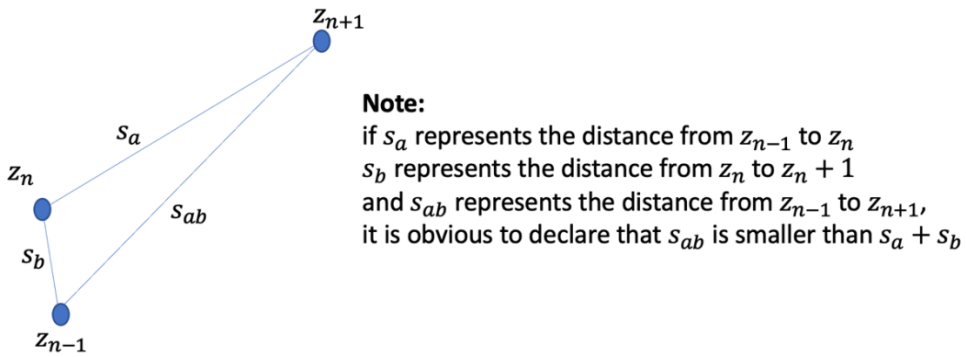

Fig. 3: The Principle of Triangular Inequality.

Algorithm 3. $Zbeacons \leftarrow \boldsymbol{PathOptimization}(\boldsymbol{T}, \boldsymbol{z_{init}}, \boldsymbol{z_{goal}})$

$Z_{beacons} = [\ \ ]$
$zn_{beacon} = z_{goal}$
$add\ zn_{beacon} to\ Zbeacons$
for $i = 2$ **to** $length(T)$ **do**
  if $obstacleFree(zn_{beacon}, T(i))$
    continue
  else
    $add\ T(i-1)\ to\ Z_{beacons}$
    $update\ cost\ of\ zn_{beacon}$
    $zn_{beacon} = T(i-1)$
  endif
  return $Z_{beacons}$
endfor

Along with getting this optimized path, the *directcost* will be calculated which will then be compared each time *pathOptimization* is performed. Once this path optimization occurs at the end of performance A* and will only repeat when the new *directcost* is better than the previous one. At the same time, when two conditions are met, n, the sampling bias variable, will be updated. Based on its magnitude and its relationship to the number of iterations, a number of samples will be generated around $Z_{beacons}$ as is the case with RRT*-Smart in general. This is intended to reduce the path again as an effort to maintain the optimality of the proposed method. Furthermore, this series of processes will be repeated until the maximum value of the iteration is met. To clarify the series of this process, Algorithm 4 is given as follows.

Algorithm 4. $\boldsymbol{RRT^{*}Smart - A*\ Algorithm}$

$T \leftarrow InitializeTree(\ )$
$T \leftarrow InsertNode(\emptyset, z_{init}, T)$
$initialPathFound = false$
for $i = 1$ **to** N **do**
  if $i = n + b, n + 2b, n + 3b \ldots$ **then**
    $z_{rand} \leftarrow intelligent\_sampling(i, z_{beacon})$
  else
    $z_{rand} \leftarrow sampling(i)$
    while $z_{rand} \in Z_{explored}$
      $z_{rand} \leftarrow sampling(i)$
    endwhile
  endif
  $z_{nearest} \leftarrow nearest(T, z_{rand})$
  $(z_{new}, u_{new}, T) \leftarrow steer(z_{nearest}, z_{rand})$
  if $obstaclefree(z_{new})$ **and** $\sim initialPathFound$ **then**
    $z_{near} \leftarrow near(T, z_{new}, |V|)$
    $z_{min} \leftarrow choosenparent(z_{near}, z_{nearest}, z_{new})$
    $T \leftarrow insertnode(z_{min}, z_{new}, T)$
    $T \leftarrow rewire(T, Z_{near}, Z_{min}, Z_{new})$
    if $z_{nearest} \in Z_{goal}$
      $T \leftarrow astart(z_{nearest}, z_{goal})$

$initialPathFound = true$
$n \leftarrow i$
endif
endif
if $initialPathfound$ **then**
$(T, directcost) \leftarrow pathOptimization(T, z_{init}, z_{goal})$
if $(directcostnew < directcostold)$**then**
$Z_{beacons} \leftarrow pathOptimization(T, z_{init}, z_{goal})$
$n \leftarrow i$
endif
endif
return T
endfor

4. RESULTS AND DISCUSSION

In this section, experimental results of different algorithms such as RRT*, RRT*-Smart, and the proposed algorithm are presented. In this experiment, the three algorithms were performed to solve the global path planning problem in some different environments that consists of a narrow channel and several barriers. They are compared to each other in terms of optimality and degree of convergence by observing the cost/distance from z_{init} to z_{goal} number of iterations, respectively.

For the first scenario, the start node is in the field area and the goal node is in a certain place. Before being tested, parameter equations were carried out to provide good observations and comparisons. This parameter includes eps which is the distance allowed to place a new node when the nearest node and a random node are given. In the first scenario experiment, eps was set to 3 for both RRT*, RRT*-Smart, and RRT*-Smart-A*. In addition, the parameter in determining the neighbor of the new node is also the same, which is set to $r_n = 3$. It should be noted that the second parameter is intended to provide equalities in the rewiring process of the three algorithms.

The next parameter equalization is the parameter used only for RRT*-Smart and the proposed algorithm, namely the beacon radius which is used for intelligent sampling performance after the initial path is found and every time the direct cost improves. In the experiment in this first scenario, the beacon radius is set equal to $r_{beacon} = 3$. In detail, the position of the start node and goal node are respectively at coordinates (5,5) and (50,65) in an environment as shown in Figure 4. The complexity of the environment is not so high, so all the algorithms compared have the ability to get the optimal path. And the optimality of this path will be observed based on the cost of each iteration and its dynamic value.

https://doi.org/10.31436/iiumej.v24i1.2529

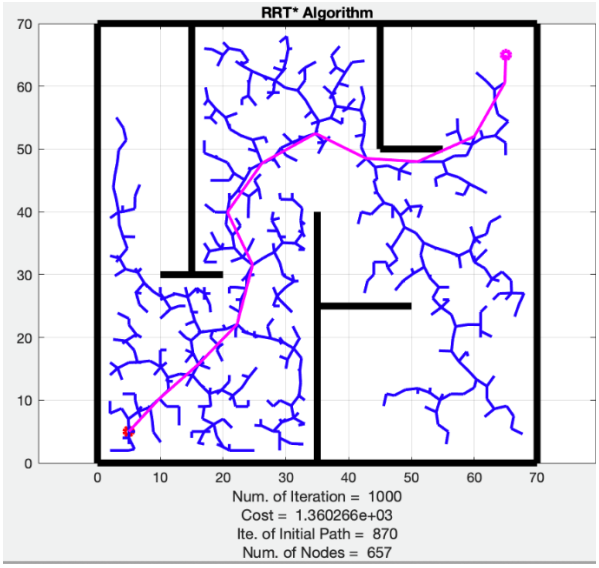

Fig. 4: Performance of RRT* Algorithm (1st Case).

Figure 4 shows a graphical result representing the performance of the RRT* Algorithm. Generally, the RRT* has succeeded in determining the collision-free path from z_{init} to z_{goal}. However, as shown in the graph and the cost value of 1360.3, the optimality of RRT* is still lacking. This is strongly relevant to the underlying theory that using simple random sampling the expansion of the nodes becomes diffuse and not centered. So, it is reasonable for a limited number of iterations, interconnecting feasible nodes is not enough to generate the optimal path.

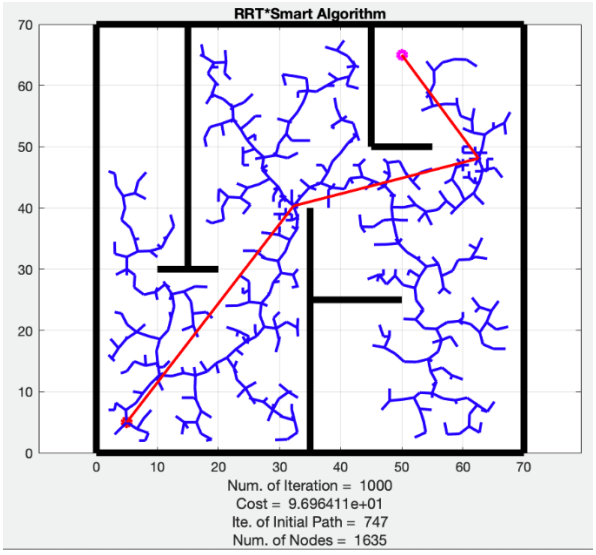

Fig. 5: Performance of RRT*-Smart Algorithm (1st Case).

In the same case, the effect of using conventional sampling on the initial path determination also makes RRT*-Smart take time to provide the optimal path. This is because the path optimization time is getting narrower when RRT*-Smart works at a limited time. This phenomenon can be seen in the need for repetition of 747 times in finding the initial path (see Fig. 5). So, it is not surprising that the final solution with a given cost of 95 can actually be increased if the application time of intelligent sampling is

https://doi.org/10.31436/iiumej.v24i1.2529

sufficient. However, this performance is enough to prove that RRT*-Smart has better optimality than RRT*.

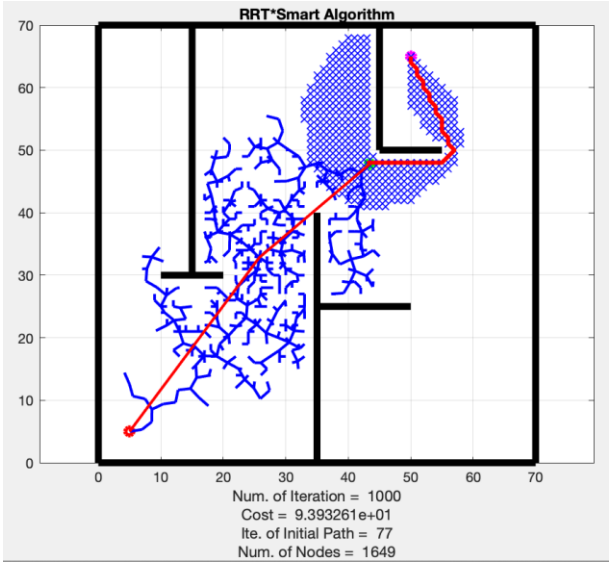

Fig. 6: Performance of RRT*Smart-A* Algorithm.

Next in this first scenario, the RRT*-Smart A* Algorithm is performed. As shown in Fig. 6, the proposed algorithms that involve fast-sampling only need 77 iterations to explore from z_{init} to Z_{goal}. This achievement is ideal because it will provide a long duration to apply intelligent sampling and path optimization to the next process. This can be seen from the distribution of samples in the area near the beacon used to assist the path optimization. Although the path optimization is not intended to repair the offered path of A, the cost is 93, it is enough to prove that in term of optimality the proposed method is better than RRT* and RRT*-Smart. Additionally, the number of nodes generated in a finite duration implies that optimization can occur iteratively, so the potential for generating shorter paths is possible.

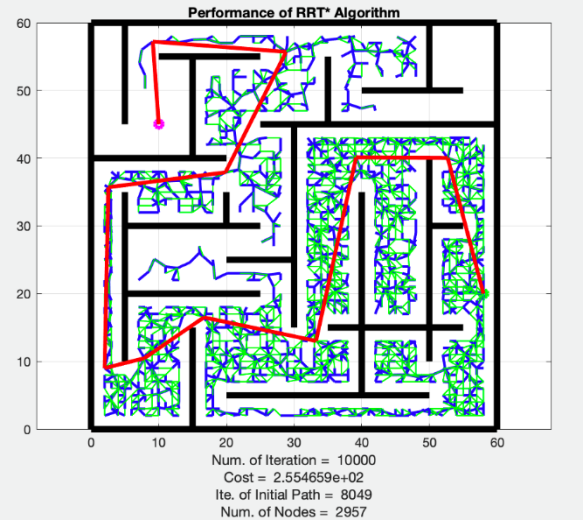

Fig. 7: Performance of RRT* Algorithm (2nd Case).

Figure 7 shows a graphical result representing the performance of the RRT* Algorithm for the second scenario. It can be seen from Fig. 7, the RRT* needs more than 8000 iterations

https://doi.org/10.31436/iiumej.v24i1.2529

to get initially optimal path before it is optimized. This situation shows that the RRT* algorithm does not have ability or is improper to solve the environment with narrow channel. Besides that, it can be seen from the path cost after optimization, the cost is high. It represents that its optimality is still lacking for this second case.

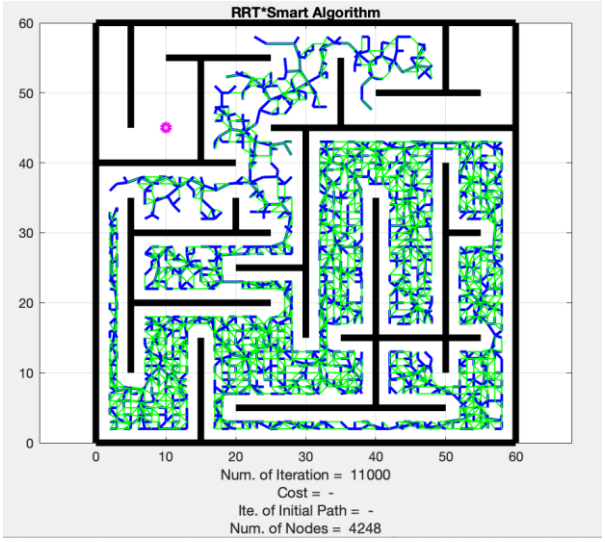

Fig. 8: Performance of RRT*-Smart Algorithm (2nd Case).

Since the conventional sampling is used for obtaining the initial path, the RRT*-smart has the similar level for its convergence speed as RRT*. It indicates that the convergence rate of RRT*-smart strongly depends on the initial sampling method. As can be seen from Fig. 8, unfortunately the RRT*-Smart does not obtain the initial path even if the number of iterations is increased and the number of nodes reaches more than 4000 nodes. For this inconsistency and limitation, the RRT*-Smart improves by replacing the conventional sampling with a fast sampling in the second case. Graphically its performance can be seen as follows.

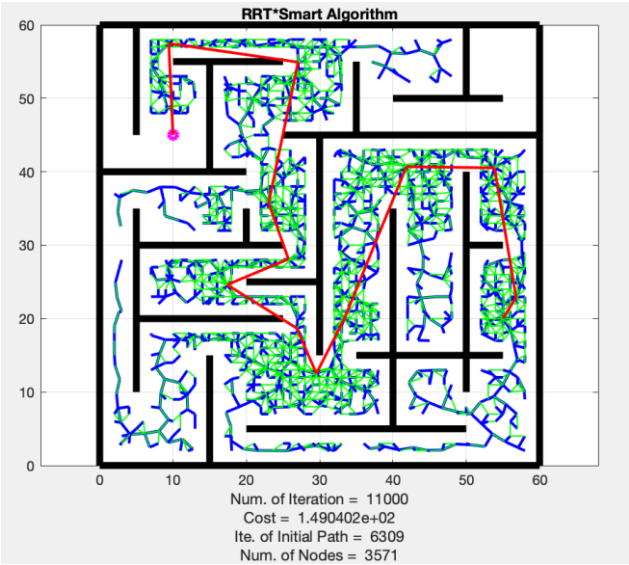

Fig. 9: Performance of RRT*-Smart with A Fast Sampling (2nd Case).

Referring to Fig. 9, the RRT*-Smart with a fast sampling is also limited. It can be observed based on the number of iterations and the iteration indicating the initial path

found. The diversity between these values represents that the optimization, which is an advantage of RRT*-Smart, has a short duration. Therefore, the generated nodes have small number compare to RRT* or its predecessor. For this reason, the proposed method is introduced and its performance can be presented as follows.

Fig. 10: Performance of RRT*-Smart-A* Algorithm (2nd Case).

It can be seen from Fig. 10, the proposed method finds the initial path in the iterations of 2643 within 6000 iterations. It indicates more than 3000 iterations that are allocated for optimization only. This small number of iterations also implies that the proposed method has a better convergence speed than its predecessor. Next, although the cost is high compared to its predecessor, it is only because of the dynamicity of the exploration process. Therefore, in terms of convergence rate and optimality, the proposed method is validated by this result.

5. CONCLUSION

Due to the limitation of RRT*-Smart when it is used in environments containing some narrow channels, the conventional sampling is improper. For this reason, it is improved by replacing the conventional sampling with A* sampling. Moreover, if a wide enough time frame is provided for its optimization, it is integrated with the A* algorithm. Henceforth, it is called An Integrated RRT*Smart-A* Algorithm. It is used to solve the global path planning problem. By comparing with its predecessors, RRT* and RRT*-Smart, the proposed method has shown better efficiency in terms of convergence rate and optimal cost.

ACKNOWLEDGEMENT

This research is supported by Universitas Mercu Buana.

REFERENCES

[1] Satapathy SC, Biswal BN, Udgata SK, Mandal JK. (2014). Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014. Advances in Intelligent Systems and Computing, 327: 175-183. https://doi.org/10.1007/978-3-319-11933-5.

[2] Adriansyah A, Suwoyo H, Tian Y. (2019). Jurnal Teknologi IMPROVING ROBOT. 3: 119-126.
[3] Tian Y, Suwoyo H, Wang W, Li L. (2019). An ASVSF-SLAM Algorithm with Time-Varying Noise Statistics Based on MAP Creation and Weighted Exponent. Mathematical Problems in Engineering, 2019(1): 1-17. https://doi.org/10.1155/2019/2765731.
[4] Al-Mutib K, Faisal M, Alsulaiman M, Abdessemed F, Ramdane H, Bencherif M. (2017). Obstacle avoidance using wall-following strategy for indoor mobile robots. 2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation, ROMA 2016, 1-6. https://doi.org/10.1109/ROMA.2016.7847817.
[5] Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). Introduction to autonomous mobile robots. MIT press.
[6] Kümmerle, R. (2013). State estimation and optimization for mobile robot navigation (Doctoral dissertation, Verlag nicht ermittelbar).
[7] Jeong IB, Lee SJ, Kim JH. (2019). Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. Expert Systems with Applications, 123: 82-90. https://doi.org/10.1016/j.eswa.2019.01.032.
[8] Baglivo L, Bellomo N, Miori G, Marcuzzi E, Pertile M, Cecco MDe. (2008). An object localization and reaching method for wheeled mobile robots using laser rangefinder. 2008 4th International IEEE Conference Intelligent Systems, IS 2008. 1. 5-6. https://doi.org/10.1109/IS.2008.4670429.
[9] Chen W, Qin H. (2011). Path planning of mobile robot based on hybrid cascaded genetic algorithm. Proceedings of the World Congress on Intelligent Control and Automation (WCICA), 501-504. https://doi.org/10.1109/WCICA.2011.5970564.
[10] Kim T, Wang Y, Sahinoglu Z, Wada T, Hara S, Qiao W. (2014). State of charge estimation based on a realtime battery model and iterative smooth variable structure filter. IEEE Innovative Smart Grid Technologies - Asia, ISGT ASIA 2014, 132-137. https://doi.org/10.1109/ISGT-Asia.2014.6873777.
[11] Mashayekhi R, Idris MYI, Anisi MH, Ahmedy I, Ali I. (2020). Informed RRT∗-Connect: An Asymptotically Optimal Single-Query Path Planning Method. IEEE Access, 8: 19842-19852. https://doi.org/10.1109/ACCESS.2020.2969316.
[12] Qureshi AH, Ayaz Y. (2016). Potential functions based sampling heuristic for optimal path planning. Autonomous Robots, 40(6): 1079-1093. https://doi.org/10.1007/s10514-015-9518-0.
[13] Perez A, Karaman S, Shkolnik A, Frazzoli E, Teller S, Walter MR. (2011). Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. IEEE International Conference on Intelligent Robots and Systems, 4307-4313. https://doi.org/10.1109/IROS.2011.6048640.
[14] Yi G, Zhou C, Cao Y, Hu H. (2021). Hybrid assembly path planning for complex products by reusing a priori data. Mathematics, 9(4): 1-13. https://doi.org/10.3390/math9040395.
[15] Chen J, Zhao Y, Xu X. (2021). Improved RRT-Connect Based Path Planning Algorithm for Mobile Robots. IEEE Access, 9: 145988-145999. https://doi.org/10.1109/ACCESS.2021.3123622.
[16] Nasir J, Islam F, Malik U, Ayaz Y, Hasan O, Khan M, Muhammad MS. (2013). RRT*-Smart: A rapid convergence implementation of RRT*,. International Journal of Advanced Robotic Systems. https://doi.org/10. 299. 10.5772/56718.
[17] Liao B, Wan F, Hua Y, Ma R, Zhu S, Qing X. (2021). F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate. Expert Systems with Applications. 184. 115457. 10.1016/j.eswa.2021.115457.
[18] Mashayekhi R, Idris MYI, Anisi MH, Ahmedy I. (2020). Hybrid RRT: A semi-dual-tree RRT-based motion planner. IEEE Access, 8: 18658-18668. https://doi.org/10.1109/ACCESS.2020.2968471.
[19] Noreen I, Khan A, Habib Z. (2016). A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms. IJCSNS International Journal of Computer Science and Network Security, 16(10): 20-27. http://cloud.politala.ac.id/politala/1. Jurusan/Teknik Informatika/19. e-journal/Jurnal Internasional TI/IJCSNS/2016 Vol. 16 No. 10/20161004_A

Compar ison of RRT, RRT and RRT - Smart Path Planning Algorithms.pdf
[20] Islam F, Nasir J, Malik U, Ayaz Y, Hasan O. (2012). RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution. 2012 IEEE International Conference on Mechatronics and Automation, ICMA 2012, 1651-1656. https://doi.org/10.1109/ICMA.2012.6284384.
[21] Noreen, I., Khan, A., Asghar, K., & Habib, Z. (2019). A Path-Planning Performance Comparison of RRT*-AB with MEA* in a 2-Dimensional Environment. Symmetry, 11(7), 945. MDPI AG. Retrieved from http://dx.doi.org/10.3390/sym11070945.
[22] Prince, C.G. (2004). Book Review: Computational Principles of Mobile Robotics. Minds and Machines 14, 407–414. https://doi.org/10.1023/B:MIND.0000035501.55990.99.
[23] Barfoot, T. (2017). State Estimation for Robotics. Cambridge: Cambridge University Press. https://doi.org/10.1017/9781316671528.
[24] Fernández-Madrigal J-A. (2012). Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods. IGI global. https://doi.org/10.4018/978-1-4666-2104-6.
[25] Gao Z, Mu D, Zhong Y, Gu C, Ren C. (2019). Adaptively Random Weighted Cubature Kalman Filter for Nonlinear Systems. Mathematical Problems in Engineering. 2019. 1-13. 10.1155/2019/4160847.
[26] Kocijan, J. (2016). Modelling and control of dynamic systems using Gaussian process models (pp. 33-38). Cham: Springer International Publishing .https://doi.org/10.1007/978-3-319-21021-6.
[27] Ni J, Wang K, Huang H, Wu L, Luo C. (2016). Robot path planning based on an improved genetic algorithm with variable length chromosome. 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD 2016, 145-149. https://doi.org/10.1109/FSKD.2016.7603165.
[28] Magzhan K, Jani H. (2013). A Review And Evaluations Of Shortest Path Algorithms. International Journal of Scientific & Technology Research, 2(6): 99-104. http://www.ijstr.org/final-print/june2013/A-Review-And-Evaluations-Of-Shortest-Path-Algorithms.pdf.
[29] Rachmawati D, Gustin L. (2020). Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem. Journal of Physics: Conference Series. 1566. 012061. https://doi.org/10.1088/1742-6596/1566/1/012061.