Robotica (2017) volume 35, pp. 1327–1340. © Cambridge University Press 2016
doi:10.1017/S0263574716000084

An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees

Andreas Bircher†, Kostas Alexis‡*, Ulrich Schwesinger†, Sammy Omari†, Michael Burri† and Roland Siegwart†

†*Autonomous Systems Lab at ETH Zurich, Leonhardstrasse 21, 8092 Zurich, Switzerland*
‡*University of Nevada, Reno, 1664 N. Virginia St., 89557, Reno, NV, US*

(Accepted January 25, 2016. First published online: March 11, 2016)

SUMMARY
A new algorithm, called rapidly exploring random tree of trees (RRTOT) is proposed, that aims to address the challenge of planning for autonomous structural inspection. Given a representation of a structure, a visibility model of an onboard sensor, an initial robot configuration and constraints, RRTOT computes inspection paths that provide full coverage. Sampling based techniques and a meta-tree structure consisting of multiple RRT* trees are employed to find admissible paths with decreasing cost. Using this approach, RRTOT does not suffer from the limitations of strategies that separate the inspection path planning problem into that of finding the minimum set of observation points and only afterwards compute the best possible path among them. Analysis is provided on the capability of RRTOT to find admissible solutions that, in the limit case, approach the optimal one. The algorithm is evaluated in both simulation and experimental studies. An unmanned rotorcraft equipped with a vision sensor was utilized as the experimental platform and validation of the achieved inspection properties was performed using **3**D reconstruction techniques.

KEYWORDS: Inspection path planning; Coverage path planning; Aerial robotics; Unmanned aerial vehicles.

1. Introduction

The vision of autonomous structural inspection operations has always inspired the research community due to its great potential in real-life applications as well as the scientific challenges it poses. Nowadays, addressing this problem efficiently gains further interest as robots have proven their capabilities in information gathering. Either in the form of unmanned ground,[1] aerial[2] or even underwater[3] vehicles and utilizing vision,[2,4,5] LiDARs[1] or other sensing devices, robots are capable of perceiving the environment and acquiring data remotely, safely and with low cost. As infrastructure ages and extends, and as the potential safety threats grow, the need for precise, frequent and time-efficient structural inspection has increased.

To address the challenges of autonomous inspection operations, a robot has to be able to compute an efficient path that results in full coverage of the structure to be inspected. This problem belongs to the class of coverage problems and despite the interest it has attracted, its inherent complications still limit the performance of the proposed solutions.

The survey in ref. [6] and the references therein provide a comprehensive overview of coverage planning methods. Some methods either simplify the inspection structure or the planning space via

* Corresponding author. E-mail: kalexis@unr.edu
E-mails: andreas.bircher@mavt.ethz.ch, ulrich.schwesinger@mavt.ethz.ch, sammy.omari@mavt.ethz.ch, michael. burri@mavt.ethz.ch, rsiegwart@ethz.ch

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

cell decomposition[7,8] or grid-based schemes.[9] More versatile approaches that can handle arbitrary 3D structures divide the global inspection path planning problem into two NP-hard subproblems, namely the art-gallery problem (AGP)[10] and the traveling salesman problem (TSP).[11] Given a visibility model, an AGP solver finds the minimal set of observation points that the robot has to visit to guarantee full coverage and subsequently a TSP solver computes the shortest route among them. The authors in refs.[3,12–14] present advanced methods that follow the separated approach. Despite their notable performance, such decoupled approaches suffer from specific limitations. First of all, they are unable to guarantee feasibility of the exact solution in case of non-holonomic constraints and cluttered environments since the points generated by the AGP solver might turn out to be unreachable from other observation points on the set. Moreover, even when a feasible path that ensures full coverage is computed, its optimality is in general not ensured due to the decoupled two-step optimization. This is because the AGP solver ignores the resulting cost of connecting the selected points. In order to alleviate these limitations in the inspection planning for complex 3D structures, new algorithms that do not separate the problem into the AGP and TSP but rather aim to compute the observation points and the path simultaneously have to be proposed. Among the first, the work in ref. [15] made a significant step towards this direction. However, its graph expansion based on sampling in the whole control space followed by forward simulations does support multiple vehicle configurations but comes at the inherent cost of computation time. Since many systems relevant to inspection (e.g. aerial robots with rotorcraft configurations) feature an explicit Boundary Value Solver (BVS), one can take advantage of recent results for point-to-point path planning like PRM* or RRT*[16] and design a fast algorithm that benefits from this fact. Following a rather hybrid approach, previous contribution of the authors aimed to make a step towards iteratively optimized (but not guaranteed optimal) solutions that retain a low computational load.[4,5] Ideally, one would desire to be able to utilize an inspection solver that can provide results with guaranteed (and if possible optimal) quality, at a computational load that allows its flexible utilization for problems of reasonable and realistic scale and complexity. Essentially, this calls for algorithms that encompass the capacity to approach the optimal solution, while at the same time being characterized by the property of computing first solutions extremely fast, and subsequently having the capacity to iteratively improve them.

Within this work, a new approach for full coverage, optimal inspection path planning, called RRTOT is proposed. For systems with a BVS, it guarantees to find inspection paths that give full coverage while, in the limit case of infinite iterations, it approaches the optimal solution. The RRTOT does not separate the problem into the AGP and TSP but rather employs a sampling-based motion-planning paradigm that computes and optimizes coverage solutions. Due to the nature of the inspection problem and in order to approach it in a unified way, a novel scheme is employed that grows a meta–tree of subtrees where each one of them expands through the whole configuration space. Algorithms using multiple trees are known from point-to-point path planning[17–19] to solve problems of parallelization, narrow passages or multi-goal path planning. However, to the best of our knowledge, it is the first time that such a meta-tree/subtree structure is employed to unify the optimization for good observation points and short paths for the coverage problem. While this structure gets incrementally built, admissible paths that provide full coverage are found. At the same time, the BVS enables fast tree-based optimization of existing paths which also reduces the time to high-quality solutions compared to ref. [15] for all these cases that a BVS is available. Vehicle limitations such as non-holonomic constraints are enforced by the use of the BVS and the resulting algorithm can be applied to all systems for which such a BVS exists.

The RRTOT algorithm is evaluated in both simulation and experimental studies for holonomic as well as non-holonomic robotic systems. For the experimental evaluation, the Micro Aerial Vehicle (MAV) platform shown in Fig. 1 is utilized and equipped with a vision sensor. Complying with the model of the onboard sensor, 3D inspection paths are computed and provided as reference to the MAV flight control unit. The completeness of the inspection results is evaluated using 3D reconstruction algorithms . A video of one of the recorded experiments can be found online at `https://youtu.be/e7ljyDM9h8o`. In all cases, the RRTOT algorithm presents satisfactory performance and computes fast and full coverage results that indicate its potential as a planner for structural inspection. Overall, the proposed algorithm employs a novel iterative strategy to approach the optimal full coverage solution while guaranteeing collision-free navigation and respecting vehicle constraints and sensor limitations. This approach is in contrast with the typically employed methodology of splitting the inspection path planning problem into that of minimal set of viewpoints

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press



Fig. 1. Instant of the RRTOT inspection path execution and derived 3D model.

computation and subsequent tour computation and path optimization. Thorough evaluation studies illustrate that such an approach not only is conceptually valuable due to its capacity to converge to the optimal solution but also practically applicable.

The remainder of this paper is structured as follows. In Section 2, the inspection problem is defined followed by the description of the proposed approach in Section 3. In Section 4, the feasibility and optimality properties of the solution are analyzed. Evaluation studies are presented in Section 5, while conclusions are drawn in Section 6.

2. Problem Definition

Consider a bounded δ-dimensional Euclidean space $X \subset \mathbb{R}^{\delta}$ ($\delta = 2$ or 3) consisting of the obstacle regions $X_{obs} \subset X$ and the resulting free space $X_{free} = X \setminus X_{obs}$. Similarly, Ξ_{free} is the free configuration space, that can be augmented with system specific states. Further, there exists a set of manifolds $M \subset X$ to be inspected. The system dynamics of the inspecting vehicle have the form $\xi_{i+1} = f(\xi_i, \mathbf{u}_i)$, where ξ is the configuration vector, $\mathbf{u}$ is the input and the initial condition is $\xi_0 = \xi_{init}$. Paths are defined as $\sigma : \mathbb{R} \rightarrow \xi$. Additionally, a visibility model is assumed, that defines the area $\mathcal{V}(\xi) \subseteq X$ that can be inspected from configuration ξ. If a manifold lies completely inside $\mathcal{V}(\xi)$, it is called visible from ξ. The inspection path planning problem consists of the triplet $(\Xi_{free}, \xi_{init}, M)$. Subsequently, the definitions of a feasible and an optimal inspection path planning problem are given following a terminology similar to the one used in ref. [16].

Problem 1 (Feasible Inspection Path Planning). Let the set of manifolds $M \subset X$ be the area to be inspected. Further, $A_i \subseteq M$, $\forall i \in [1, N]$, with $N \in \mathbb{N}^+$ and $\bigcup_{i=[1,2,\ldots N]} A_i = M$. Given an inspection path planning problem $(\Xi_{free}, \xi_{init}, M)$, find an *admissible path* $\sigma(s)$, $s \in [0, 1]$ such that $\sigma(0) = \xi_{init}$ and $\forall i \in [1, N] \exists s_i$ such that $\sigma(s_i)$ is a configuration from which A_i is visible. If such a path exists, the problem is feasible.

Problem 2 (Optimal Inspection Path Planning). Let Σ be the set of all admissible paths. Given a feasible inspection path planning problem $(\Xi_{free}, \xi_{init}, M)$ and a cost function $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$, find an admissible path σ^* such that $c(\sigma^*) = min_{\sigma \in \Sigma} c(\sigma)$.

3. Proposed Approach

A characteristic difference of point-to-point path planning and inspection (coverage) path planning is that in the latter, once a specific subset of the structure has been inspected, paths may turn towards a new target and follow this new general direction that can even be the same way back. Reflecting this fact and in order to make use of recent results for point-to-point path planning that allow rapid

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

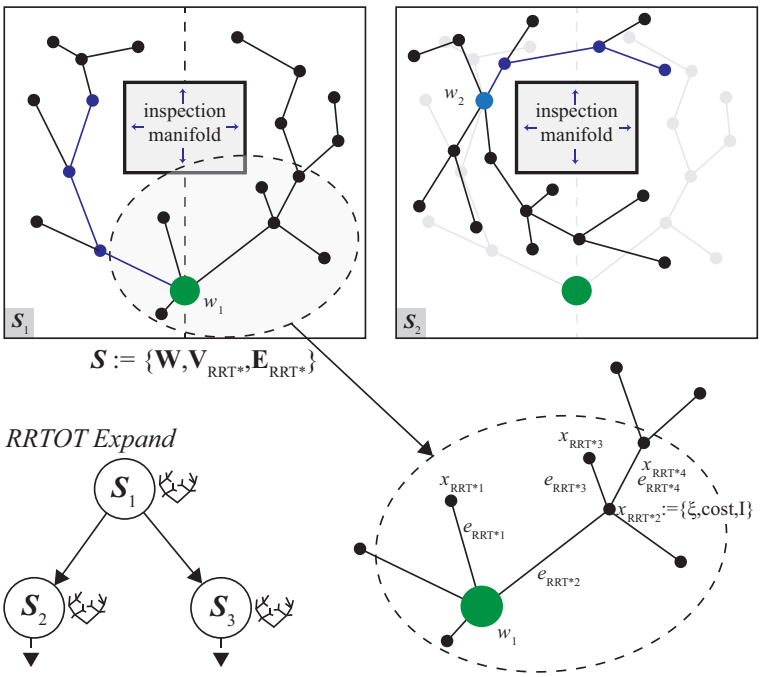


Fig. 2. Starting from ξ_{init}, the initial RRT* tree grows to cover Ξ_{free}. Of its vertices, the root configurations for new subtrees are randomly sampled. The new roots become waypoints in the new subtrees. By iteratively sampling new waypoints from existing RRT* trees, a TOT grows to find arbitrary complex paths.

exploration of the configuration space, our approach proposes a division of inspection paths into segments which follow a certain direction, that may change when switching to a new segment. From potential switching points, called *waypoints*, several path segments may branch. The resulting directed graph has a tree structure, which allows extracting solutions easily. On a lower level, each waypoint is a root of a RRT* tree, called subtree. Such a subtree spans the whole free configuration space. The end points of the next path segments are chosen from its vertices, effectively resulting in the choice of a new direction and distance to travel. This meta-tree, called tree of trees (TOT), is constructed randomly and incrementally, as outlined in Algorithm 1 and the according function descriptions. A key feature of the proposed algorithm is the optimization of existing paths (Algorithm 1, lines 9 and 12) while the TOT is constructed. Aforementioned structure allows for local cost optimization in the subtrees, while searching for full visibility paths can be mostly performed on the level of the TOT.

More formally, the TOT consists of its subtrees $S \in \mathbb{S}$, $S := \{\mathbb{W}^S, \mathbb{V}^S, \mathbb{E}^S\}$ ($\mathbb{S}$ represens the set of subtrees), consisting of waypoints $w \in \mathbb{W}^S$, vertices $x \in \mathbb{V}^S$ and edges $e \in \mathbb{E}^S$. Each vertex x is annotated with a configuration of the vehicle ξ, a cost and a set $I \subseteq M$ that is the set of the already inspected areas. It is the union of the visible areas along the path $I(\sigma(t)) = \bigcup_{s=[0,t]} \mathcal{V}(\sigma(s)) \cap M$. To construct these vertices and edges, the algorithm employs the RRT* method[16] that grows the subtrees in N_{RRT^*} iterations. Every subtree possesses its own RRT* planner that has no knowledge of the other subtrees. A new waypoint is chosen from the set of vertices of its respective parent subtree, to which it will be connected. This effectively maintains a global tree structure as shown in Fig. 2. To eventually achieve full coverage, the depth of the TOT grows with increasing iterations to find more complex paths that give higher visibility.

The annotation of vertices with the set of the already inspected part of the structure I enables the extraction of actual solutions. Every time a vertex reaches full visibility ($I = M$) and the cost is lower than the *best cost* so far, the path to this vertex is the new *best path* so far $\hat{\sigma}$. The latest best path is available as the output of the algorithm at any time.

Once a first solution is found, more effort is dedicated into improving existing solutions. As shown in Algorithm 1, at the end of every TOT-iteration, every subtree's RRT* planner is iterated for $N_{optimize}$ times. In order not to destroy existing admissible paths, the RRT* rewiring criterion has to be adapted: a vertex is only rewired, if besides decreasing the cost it does not lose any visibility,

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

that is $I \subseteq I'$, where I is the original and I' the new visibility. Additionally, the waypoint positions are also optimized in a rewiring step as shown in Algorithm 2. In this step, subtrees may accumulate multiple waypoints that are roots of the same subtree, that is then actually a forest (set of disjoint trees). In the optimization's rewiring step, two trees in the forest may get reconnected, effectively removing a root. Consequently, the waypoint that previously was this root is also removed.

3.1. Subroutines

The role of the employed functions within Algorithms 1 and 2 is briefly explained below:

NewSubtree(ξ, N_{RRT^*})**:** A standard RRT* tree with N_{RRT^*} iterations is grown, starting at ξ as root. If a vertex gets full visibility, $I = M$ and has lower cost from root (*CFR()*) than the best cost $c(\hat{\sigma})$, the path to the vertex is the new global best path $\hat{\sigma}$. The output is this new subtree S.

ExpandSubtree(S)**:** This function samples a vertex at random from $\mathbb{V}^S$ to use its configuration to initialize a new subtree (NewSubtree()). The same vertex can only be sampled once.

OptimizeSubtree(S, $N_{optimize}$)**:** This routine iterates the RRT* planner of the subtree S for $N_{optimize}$ iterations. The condition in the rewiring step is augmented with the requirement, that the near vertices must not lose visibility.

Near(S, x)**:** The output of this function is the set of all vertices in $\mathbb{V}^S$ that are within a distance d of x according to some distance metric using the vertices configurations. This distance d scales with $\frac{log(n)}{n}$, where n is the number of vertices in $\mathbb{V}^S$. The used distance threshold is the same as for the RRT* Near() function which is detailed in ref. [16].

Algorithm 1: This algorithm presents the body of the RRTOT inspection strategy. It iteratively builds the TOT and as soon as a first solution is found, optimizes existing paths at the end of every iteration. The functions *NewSubtree*, *OptimizeSubtree* and *RewireWaypoint* update the globally best solution $\hat{\sigma}$.

$\mathbb{S} \leftarrow S_{init} \leftarrow$ NewSubtree(ξ_{init}, N_{RRT^*})
$N_{\mathrm{TOT}} \leftarrow 0$
while execute **do**
  for all $S \in \mathbb{S}$ **do**
    $\mathbb{S} \leftarrow \mathbb{S} \cup$ ExpandSubtree(S)
  end for
  if first admissible path found **then**
    for all $S \in \mathbb{S}$ **do**
      OptimizeSubtree(S, $N_{optimize}$)
    end for
    for all $S \in \mathbb{S} \setminus S_{init}$ **do**
      RewireWaypoint(S)
    end for
  end if
  $N_{\mathrm{TOT}} \leftarrow N_{\mathrm{TOT}} + 1$
end while
return best admissible path $\hat{\sigma}$

ObstacleFree(x_i, x_j)**:** This function returns true, if the path from ξ_{x_i} of x_i to ξ_{x_j} of x_j lies entirely in Ξ_{free}. The path is subject to the encoded vehicle constraints.

CFR(x)**:** The CFR function returns the cost of moving from ξ_{init} to ξ_x of x. The path considered for the cost calculation runs along the edges of the TOT and its subtrees.

VFR(x)**:** The Visibility From Root (VFR) function returns the union of the visibility along the edges of the TOT and its subtrees from ξ_{init} to ξ_x, the configuration of x, which corresponds to I of x.

Cost(x_i, x_j)**:** The cost of moving (typically time-to-travel) from one vertex x_i to another, x_j is returned. The connecting path is subject to the vertices' configurations ξ_i and ξ_j and the system constraints.

Visibility(x_i, x_j)**:** The function returns the set of visible manifolds when moving from x_i to x_j, $\bigcup_{s=[0,1]} \mathcal{V}(\sigma(s)) \cap M$, with $\sigma(0) = \xi_i$, $\sigma(1) = \xi_j$, where ξ_i, ξ_j are the configurations corresponding

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

to vertices x_i, x_j respectively. Explicit solutions for the computation may exist for some systems, otherwise discrete sampling of the configurations along the path can be used as an approximation.
Parent(x)**:** Returns the parent vertex of x. Note that this may be part of a different subtree.

Algorithm 2 *RewireWaypoint*(S)**:** This algorithm presents the routine, that performs a rewiring step for the waypoints. It improves the paths in these regions, which are not affected by the optimization of the *OptimizeSubtree* function. It is highlighted that a check that confirms weather rewiring of the x_{child} with x_{near} would result in a lower cost compared to the current cost from root, while none of the VFR is lost (but could potentially be complemented by additional visibility) is performed. Only if this is the case and the edge is collision free, then the rewiring step is performed.

1: $\mathbb{V}_{children} \leftarrow$ Get child vertices of all waypoints $\mathbb{W}^S$
2: **for** *all* $x_{child} \in \mathbb{V}_{children}$ **do**
3: $S_{parent} \leftarrow$ Get parent subtree of S;
4: $\mathbb{V}_{near} \leftarrow$ Near(S_{parent}, x_{child});
5: **for** *all* $x_{near} \in \mathbb{V}_{near}$ **do**
6: **if** ObstacleFree(x_{near}, x_{child}) & CFR$(x_{child}) \geq$ CFR(x_{near}) + Cost(x_{near}, x_{child}) [4] & VFR$(x_{child}) \subseteq$ VFR$(x_{near}) \cup$ Visibility(x_{near}, x_{child}) **then**
7: $\mathbb{W}^S \leftarrow \mathbb{W}^S \cup \{x_{child}\}$;
8: $x_{parent} \leftarrow$ Parent(x_{child});
9: $\mathbb{E}^S \leftarrow \mathbb{E}^S \setminus \{(x_{parent}, x_{child})\}$;
10: $\mathbb{E}^S \leftarrow \mathbb{E}^S \cup \{(x_{near}, x_{child})\}$;
11: VFR$(x_{child}) \leftarrow$VFR$(x_{near}) \cup$ Visibility(x_{near}, x_{child});
12: **if** VFR$(x_{child}) = M$ & $c(\hat{\sigma}) >$ CFR(x_{child})
13: $\hat{\sigma} \leftarrow$ GetPath(x_{child});
14: **end if**
15: **end if**
16: **end for**
17: **end for**

GetPath(x)**:** This function reconstructs the path along the edges of the TOT and its subtrees from ξ_{init} to ξ_x, the configuration of x. This path σ is returned.

3.2. Parameters

The proposed algorithm employs a small set of tuning parameters, an aspect particularly important for its straightforward utilization in real-life operations. These are briefly defined below:

N_{RRT^*}: Defines the number of iterations on the level of the RRT* planner in the subtrees. It is relevant for the initial computation of the tree. A high number generally results in a slower first solution with a better quality, while a lower number of iterations increases the chance of a fast first solution. In either way, it should be chosen high enough to cover the whole Ξ_{free}.
N_{optimize}: Defines the number of iterations on the subtree-level for the optimization step (Algorithm 1, line 9). Typically, smaller values than N_{RRT^*} are used.

4. Analysis

This section deals with the formal analysis of the full coverage feasibility and inspection path optimality characteristics of the proposed approach. Within this analysis, specific assumptions are made and firstly that the RRTOT uses RRT* planners without limitation of the edge length, thus connecting with a newly sampled vertex exactly. In addition, several assumptions about the problem setup have to be made for the analysis but also for the algorithm. Most importantly, the requirements for the RRT* algorithm[16] have to be fulfilled, namely the continuity and additivity of the cost function. In addition, the following assumptions are made:

Assumption 1 (Uniform Sampling). The sampling procedure is such that the samples are drawn from an absolutely continuous and uniform distribution on Ξ_{free}.

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

Assumption 2 (Initial RRT* Iterations). It is assumed, that the number of initial iterations N_{RRT^*} is not small, such that samples in regions that cannot be connected directly to root because of obstacles can be connected via previously sampled vertices and have a probability to exist after N_{RRT^*} iterations that is arbitrarily close to the obstacle free case. In addition, the largeness of N_{RRT^*} is such, that when sampling new waypoints from the vertices, the samples can be assumed independent.

Assumption 3 (Inspectability). Each piece of inspection structure $\{A_1, A_2, \ldots A_N\}$ has a nonempty open area $B(A_i) \subseteq \Xi_{free}$, called inspectability area, from which it can be inspected. That is $\mu(B(A_i)) > 0$, where $\mu(\Xi)$ denotes the Lebesgue-measure.

The aforementioned strong assumptions are necessary for the formal proofs that follow. It is however worth mentioning, that as the experimental sections will show, the algorithm behaves in a very efficient way even when aspects of these assumptions cannot easily be proven to hold in an exact sense for the experimental data, the limitations of the computational system and the constraints of the vision sensor.

4.1. Feasibility of the algorithm

Subsequently, we present the proof, that the RRTOT algorithm finds an *admissible path* with a probability that *asymptotically converges to* 1 as $N_{RRT^*} \to \infty$. In a first step, it is shown, that long enough paths are found and afterwards a lower bound for the probability to find a solution is established, that converges to 1 as the number of waypoints in a path increases.

Theorem 1 (Existence of Collision-free Long Paths). As the number of iterations $N_{TOT} \to \infty$, so does the number of waypoints r in the longest sequence of consecutive waypoints P for a feasible problem.

Proof. As the number of iterations $N_{TOT} \to \infty$, the root waypoint $w_1(\xi_{init})$ has infinite children waypoints, since in every iteration one is added. For a feasible problem, in each iteration N_{TOT} the probability of sampling a configuration $\xi \to w_2$ that, immediately or later, results in collision is $p_{N_{TOT}} < 1$ and the probability of not finding a collision-free w_2 decays to zero as $N_{TOT} \to \infty$, $\prod_{i=1}^{\infty} p_i = 0$. The same holds for any waypoints w_k and w_{k+1}, $k \in \mathbb{N}^+$. From $r > k$ follows $r \to \infty$. □

Definition 1 ((α, β)-expansiveness[20]). Let $\mathcal{R}(p)$ and $\mathcal{R}_l(p)$ be the reachable set and the locally reachable set that is reachable in one expand step, respectively. The reachability of a set $Q \subset \Xi_{free}$ is $\mathcal{R}(Q) = \bigcup_{p \in Q} \mathcal{R}(p)$ and the local reachabilty of Q is $\mathcal{R}_l(Q) = \bigcup_{p \in Q} \mathcal{R}_l(p)$. Let β be a constant in $(0, 1]$, then the β-lookout of the set Q is defined as β-LOOKOUT$(Q) = \{p \in Q \mid \mu(\mathcal{R}_l(p) \setminus Q) \geq \beta\mu(\mathcal{R}(Q) \setminus Q)\}$. Let α be a constant in $(0, 1]$. The free space Ξ_{free} is (α, β)-expansive if for every point $p \in \Xi_{free}$ and every subset $Q \subset \mathcal{R}(p)$, it holds that $\mu(\beta\text{-LOOKOUT}(Q)) \geq \alpha\mu(Q)$.

In the RRTOT algorithm, a new waypoint can be sampled anywhere in Ξ_{free} without limitations by Assumption 2. Therefore, the (local) reachability of the system is always the whole free space Ξ_{free}. Thus, for the RRTOT algorithm the free space is (α, β)-expansive, in particular $\alpha = \beta = 1$.

Theorem 2 (ref. [20]). Let Ξ_{free} be (α, β)-expansive, $g_i > 0$ be the volume of the inspectability area $B(A_i)$ and γ_i be a constant in $(0, 1]$. A sequence P of r waypoints contains a waypoint in $B(A_i)$ with probability at least $1 - \gamma_i$, if

$$r \geq (k_i/\alpha) ln(2k_i/\gamma_i) + (2/g_i) ln(2/\gamma_i) = r_{min,i}, \tag{1}$$

where $k_i = (1/\beta) ln(2/g_i)$.

The main idea for the proof of Theorem 2[20] is to calculate the probabilistic overlap of the inspectability area with the local reachability set of the sequence P. This gives the probability to sample a waypoint in the next iteration that lies inside $B(A_i)$. The expression in Eq. (1) gives the bound on that probability. This is extendable to multiple inspectability areas that are visited in a row. Doing so provides the result required.

Corollary 1 (to Theorem 2). A sequence P of r waypoints contains a waypoint in each $B(A_i)$, $\forall i \in \{1, 2, \ldots N\}$ with probability at least $\prod_{i=1}^{N}(1 - \gamma_i)$ if $r \geq \sum_{i=1}^{N} r_{min,i}$. Moreover, if the number

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

of iterations $N_{TOT} \to \infty$, then $r \to \infty$ as shown in Theorem 1. Suppose P is partitioned such that $r_{min,i} = r_{min,j}$, $i \neq j$ and the length of the subsequences goes to infinity as well. All γ_i decay to zero exponentially fast and an admissible path for problem 1 is found *almost surely*, that is $\lim_{r\to\infty} \prod_{i=1}^{N}(1-\gamma_i) = 1$. Therefore, RRTOT finds an *admissible path* with probability that converges to 1.

The key parts for showing feasibility of the algorithm were the relation of the lower bound of the probability to find a solution with the length of a sequence given by Theorem 2. Theorem 1 guarantees, that long enough sequences will actually be generated for the probability to converge to 1. The proof of finding full coverage solutions corresponds to the most fundamental property of the proposed algorithm.

4.2. Asymptotic optimality

Asymptotic optimality is a particularly challenging property for inspection path planning algorithms. Within this section, it is shown that a *baseline* version of the RRTOT algorithm turns out to express such a critical property. More specifically, the algorithm considered for the analysis of asymptotic optimality excludes the waypoint rewiring function (Algorithm 1, line 12) and is called the *basic-RRTOT*. Experience gathered in simulation and experiments showed that the waypoint rewiring process only benefits the algorithm evolution towards good solutions due to its intrinsic criterion (Algorithm 2, line 6). In general, solutions of the *basic-RRTOT* are observed to be therefore equal or worse than the real RRTOT's solutions that has been executed for the same amount of iterations. However, formal analysis of asymptotic optimality is only provided for the *basic*-RROT, a structurally simplified algorithm that can be used for theoretical comparisons and evaluation of other methods. In particular and under specific assumptions, it is shown that in the limit of $N_{TOT} \to \infty$, the set of admissible paths found by the *basic-RRTOT* are probabilistically equivalent to the solutions found by an idealized algorithm that iteratively executes an *ideal-Expand* function (Algorithm 3) on a tree consisting of waypoints and edges. While expanding, the algorithm keeps track of the inspected part of the structure for each waypoint. For this algorithm, the convergence to the optimal solution is proven in ref. [15]. This proof shows that the algorithm can sample a series of waypoints arbitrarily close to the optimal path in the limit of infinite iterations. As the *basic-RRTOT* finds the same solutions also one arbitrary close to the optimal will be among them. The necessary assumptions for the comparison are:

Assumption 4 (ϵ-Inspectability). There exists a constant $\epsilon \in \mathbb{R}^+$ and a suboptimal path $\sigma'(s)$, $s \in [0, 1]$, that follows the optimal path σ^* arbitrarily close, and the respective costs are arbitrarily close, such that for every A_i $\exists s_i$ with $\mathcal{B}_{\sigma'(s_i),\epsilon} \subseteq B(A_i)$, where $\mathcal{B}_{\sigma'(s_i),\epsilon}$ is the ϵ-ball centered at $\sigma'(s_i)$.

Assumption 4 rules out cases, where convergence towards the optimum is impossible, because no path close to the optimal solution is admissible.

The modification of the RRT* rewire criterion for the optimization might in some cases prevent the RRT* planners from converging to optimal paths within the subtrees. Whether this is the case or not depends on the order of sampling of the vertices but not on the set of vertices. The following assumption is therefore reasonable and necessary for the analysis of convergence to hold:

Assumption 5 (Non-optimality of Modified RRT*). Let $\xi \in \Xi_{free}$ be a configuration where an RRT* tree is initialized. When choosing a new waypoint w among its vertices, there is a non-zero probability, independent of its configuration ξ_w, that the connection of ξ to ξ_w will converge to the optimal as the RRT* iterations go to infinity.

The following analysis considers only the subset of optimal subtrees $\mathbb{S}_o \subseteq \mathbb{S}$, whose waypoints are optimally connected according to assumption 5 and are not disjoint. By Assumption 5, $\mathbb{S}_o$ grows infinitely deep and wide, as $N_{TOT} \to \infty$.

Theorem 3 (Asymptotic optimality of RRTOT). Under assumption 5, the solutions of the *basic-RRTOT* algorithm are the same as the solutions of the algorithm with the ideal-expand algorithm proven to lead in optimal solutions as shown in ref. [15].

Proof. This proof shows that the steps performed by idealexpand are equivalent to the steps performed by *basic-RRTOT*, from which follows that the solutions will be probabilistically the same.

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

Algorithm 3: Ideal-Expand($\mathbb{W}$)[15]

1: Sample a waypoint w' uniformly at random from $\mathcal{R}_l(\mathbb{W})$.
2: Sample a waypoint w that can reach w'.
3: Compute a control input u to move from w to w'.
4: **return** w', w, $Trajectory(w, w', u)$, u

Since $\mathcal{R}_l(w) = \Xi_{free}, \forall w \in \mathbb{W} = \mathbb{W}_{S_1} \cup \mathbb{W}_{S_2} \cup \ldots$, the order of step 1 and step 2 in Algorithm 3 is irrelevant, because every existing waypoint w can be connected to any newly sampled w'. Step 2 in the ideal-expand is equivalent to choosing the subtree to expand in the *basic-RRTOT*. Ideal-expand assumes sampling from the waypoints uniformly at random over Ξ_{free}. Again since $\mathcal{R}_l(w) = \Xi_{free}$, the waypoints are uniformly random and independent samples. In one iteration, the set of all waypoints is the set to be expanded, thus guaranteeing uniform randomness and independence. On the other hand, the vertices in an RRT* tree are also uniformly randomly distributed on Ξ_{free} and independent. Therefore, sampling new waypoints from the large set of vertices $\mathbb{V}^S$ at random is equivalent to step 1 in ideal-expand. Step 3 is performed by the RRT* in the presented algorithm, that plans the path in the configuration space and under Assumption 5 for the $OptimizeSubtree()$ function ensures the asymptotic convergence of the edges of the TOT to the optimal. From the equivalence of the two algorithms follows the equivalence of their solutions in the limit of the number of iterations going to infinity and hence the same optimality properties. □

4.3. RRTOT fast inspection solutions

The complete RRTOT algorithm (including the waypoint rewiring step) relies on specific properties and steps to provide fast inspection solutions. This section deals with an overview of the key mechanisms that enable the fast computation of full coverage inspection solutions. First of all, the (1, 1) expansiveness of the algorithm on Ξ_{free} easily makes up for the computational burden of generating RRT* trees to connect the waypoints. Especially when planning with obstacles, it is beneficial to be able to plan with connections around them, the cost of which decreases by keeping iterating the RRT* planners in the $OptimizeSubtree()$ function. Furthermore, with the rewiring of the waypoints ($RewireWaypoints()$) optimization along the whole path is achieved. The waypoints are no longer fixed points on the path that may be suboptimal and therefore would prevent the specific path from converging to the optimal. Together with the fast expansion, these two functions, that improve existing paths, result in an algorithm that finds first solutions quickly, while having the tendency to optimize them. This is particularly useful for real-life operations, where a first path is preferred to be available in short.

5. Case Studies

In order to thoroughly evaluate the capabilities of the proposed RRTOT algorithm, several simulation and experimental studies were conducted. The results verify the functionality of the algorithm and its capabilities to compute efficient inspection paths in both 2D and 3D environments for holonomic or non-holonomic vehicles. In all cases, the objective is to provide full visibility while minimizing the distance. All calculations were performed on a computer with a 3.3 GHz processor running Ubuntu 12.04 and using a single-thread C++ implementation.

5.1. 2D simulation studies

In a 2D study, the RRTOT algorithm is evaluated in indirect comparison to the performance of the algorithm[15] using an identical simulation setup as the one employed by the authors of this work. The space configuration of this simulation setup is depicted in Figs. 3 and 4. As shown, it consists of five rectangles the edges of which have to be completely inspected while the overall distance of the inspection path gets minimized. Two cases are investigated, one considering a holonomic vehicle, while a non-holonomic vehicle with a minimum turning radius of 2 units and non-negative velocity is employed for the second case. For the holonomic case, the configuration vector is $\xi^{v,H} = [x, y]^T$ with initial conditions $\xi_0^{v,H} = [45, 0]^T$, while for the non-holonomic case, the configuration vector is augmented with the vehicle's heading angle ($\xi^{v,NH} = [x, y, \psi]^T$ and $\xi_0^{v,NH} = [45, 0, \pi/2]^T$).

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

Table I. RRTOT Performance analysis.

Holonomic	1–st sol.	$c(\hat{\sigma}) < 185$ units	$c(\hat{\sigma}) < 169$ units
$\bar{t}_H$	0.25 s	89 s	336 s
$std(\bar{t})_H$	0.27 s	189 s	247 s
Non-holonomic	1–st sol.	$c(\hat{\sigma}) < 185$ units	$c(\hat{\sigma}) < 177$ units
$\bar{t}_{NH}$	3.3 s	559 s	28 min
$std(\bar{t})_{NH}$	2 s	795 s	29 min

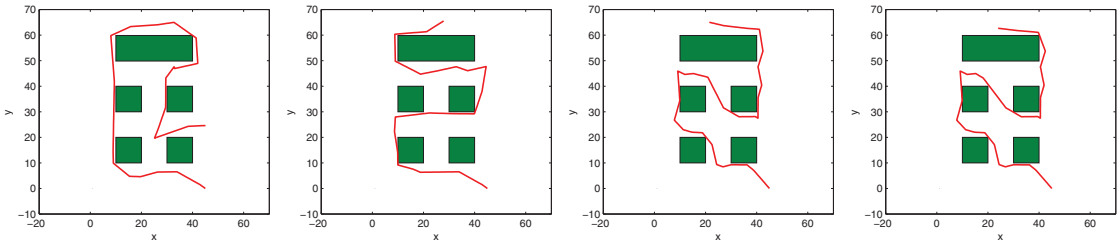


Fig. 3. A sample run of the RRTOT algorithm for the holonomic vehicle employing an omnidirectional sensor with a limited range of 15 units as only constraint ensuring inspection quality. The first result is returned quickly. The algorithm subsequently finds other paths belonging to different topologies. Figures 3c, 3d show, that a path is optimized within a certain topology. (a) Cost 196.4 in 1.1 s. (b) Cost 174.6 in 49.8 s. (c) Cost 166.7 in 140.3 s. (d) Cost 160.7 in 249.1 s.

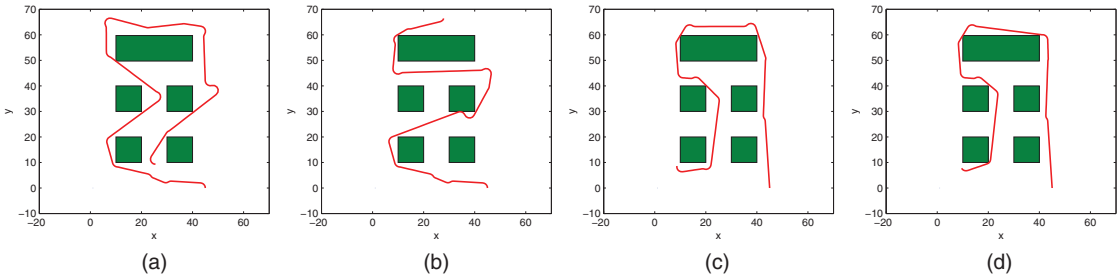


Fig. 4. A sample run of the RRTOT algorithm for the non-holonomic vehicle with a minimum turning radius of 2 units. It employs an omnidirectional sensor with a limited range of 15 units as only constraint ensuring inspection quality. The first result is returned quickly. The algorithm subsequently finds other paths belonging to different topologies. Figures 4c, 4d show, that a path is optimized within a certain topology. (a) Cost 230.6 in 2.8 s. (b) Cost 182.1 in 55.5 s. (c) Cost 176.4 in 305 s. (d) Cost 167.5 in 673 s.

Regarding the visibility model, both vehicles are considered to be equipped with an omnidirectional sensor with a range of 15 units, unless areas are occluded by parts of the structure.

Figures 3 and 4 show the computed paths in sample runs for the holonomic and non-holonomic case respectively. Within them, Figs. 3a and 4a show the first fast initial full coverage paths. In the course of computation, the algorithm comes up with better solutions, belonging to different topologies as shown in Figs. 3b, 3c and 4b, 4c. Figures 3d and 4d show that the paths are optimized within a certain topology to find solutions of decreasing cost while retaining full visibility.

To gain a statistical insight on the algorithm performance characteristics, the aforementioned process is repeated in total 20 times. The results are summarized in Table I, while the cost plots showing the convergence are depicted in Fig. 5. For further evaluation, the algorithm in ref. [15] finds, for the identical inspection problem setup, a first solution to the same non-holonomic problem in 12.87 s (standard deviation of 25.4 s) and high quality solutions with lengths of 190 and 185 units after 44 and 73 min respectively (with standard deviations of 53 and 85 min). It is highlighted once more that this comparison is indirect as the algorithm in ref. [15] can handle vehicle configurations that do not have a BVS and therefore is more versatile regarding the robot configuration. Nonetheless, it illustrates the good performance characteristics of RRTOT for all these cases of vehicles that a

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

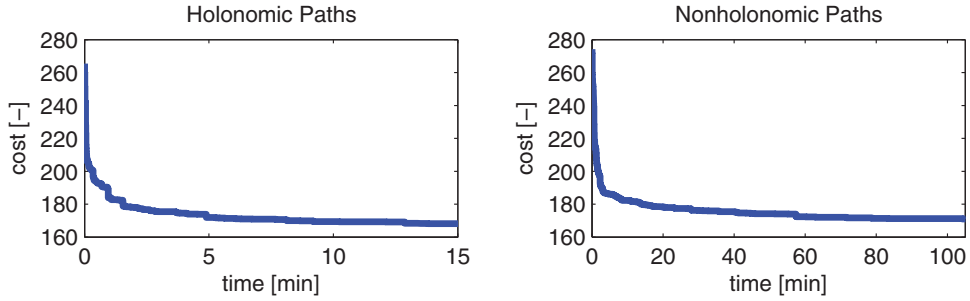


Fig. 5. Convergence curves of the 2D case study. Both curves show a fast first solution with subsequent steep convergence. For longer runtimes, the curve still exhibits decreasing cost.

BVS is available. As a very wide set of robots such as multirotor MAVs, most of the remotely operated underwater vehicles and other vastly used vehicle configurations are considered to have BVS describing their motion characteristics, RRTOT is believed to have a great potential for real-life applications.

5.2. 3D experimental studies

The experimental studies were conducted using an AscTec Firefly hexarotor MAV equipped with an Intel core duo computer and the visual–inertial sensor (VI-sensor) developed by the autonomous systems lab and Skybotix AG. The VIsensor integrates two HDR global shutter cameras (Aptina MT9V034) and an analog devices ADIS16448 IMU in a tightly aligned and synchronized way using an Artix^{TM}–7 FPGA, a Xilinx Zynq 7020 SoC module and an ATOM CPU running Linux. This integrated sensor system runs advanced image processing algorithms, provides complete pose estimates and builds a 3D map of the environment. An overview of the experimental platform is indicated in Fig. 6, while detailed information on the platform, the onboard controllers, the perception modules and the state estimation strategies are found in refs. [2,21–23]. For the experiments presented, a Vicon motion capture system provided exact position estimates and the Robot Operating System (ROS) was utilized as the middleware to allow solid interfacing of the RRTOT algorithm with the MAV. Precomputed paths are loaded to the MAV to be executed by the autopilot. As the recorded flight response is expected to naturally present a tracking error, for all the path computations the sensor FoV was considered slightly reduced compared to its actual value in order to provide some necessary robustness and avoid cases of not inspecting a subset of the desired structure. In general, as the proposed algorithm solves the problem of explicit inspection path planning, it naturally cannot *per se* with uncertainties regarding the model structure or inaccuracies of the sensor and vehicle model. To provide robustness, the main considerations of assuming a slightly reduced FoV and possibly more constrained non-holonomic constraints or slower maximum yaw rates are proposed.

The experimental setup refers to an area of 4 m $\times$ 4.5 m where multiple carton boxes are deployed and arranged as depicted in Fig. 7. Within this scenario, all non-overlapping surfaces of the boxes have to be inspected with minimal distance taking into account the limitations of the camera system (a single camera model is considered). Consequently, the visibility model is adapted to match the properties of the real sensor and therefore the field of view is angularly constrained on all four sides. From the nominal field of view a margin is deducted on all sides to account for errors of the inspection manifold model and the tracking of the path. A minimum and maximum distance to the object under inspection is also enquired to ensure good quality of inspection results. The orientation of the camera in the xy plane is an additional system state. Finally, although the employed vehicle can fly holonomic trajectories, the experiment is performed under both holonomic and non-holonomic considerations in order to evaluate the capabilities of the proposed inspection path planner. The encoded non-holonomic constraints are a minimum turning radius of $r_{\min} = 0.25$ m on the xy plane and first-order differentiability of the height. Such a setup is suitable for a multirotor MAV. With a bound on the height derivative and a larger radius constraint, it could also be adapted to plan paths for fixed-wing airplanes with a camera, movable around the z-axis. Figures 7a and 7b depict the offline computed paths for the holonomic and the non-holonomic case along with the experimentally

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

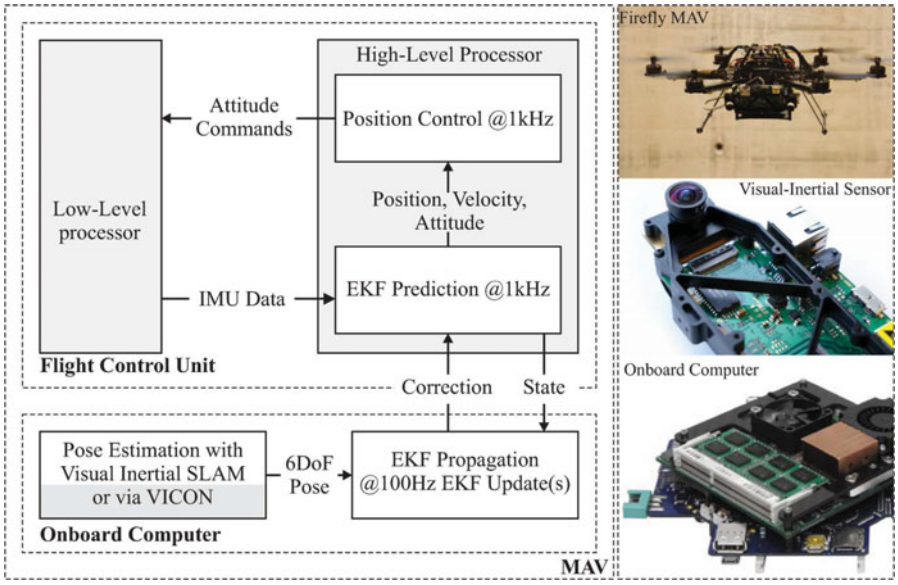


Fig. 6. Block diagram of the main algorithmical components of the flight control unit of the Firefly hexarotor, photos of the vehicle, the VI-Sensor and the high-level core duo processor. The lowlevel processor runs the attitude control of the vehicle which is commanded by the position controller running the high-level processor. The high-level unit also executes the state estimation algorithms and iterfaces the VICON pose feeds.

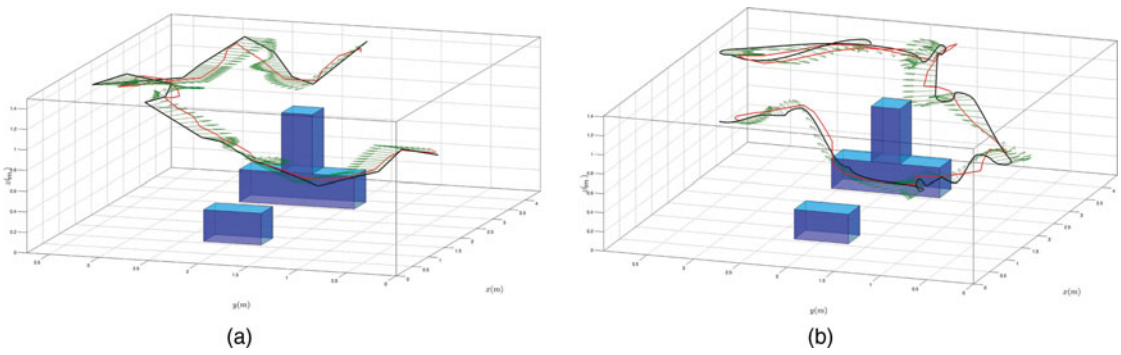


Fig. 7. Inspection paths computed by the RRTOT planner (black) and plots of the recorded vehicle response (red). The considered vehicle employs a vision sensor with a limited field of view and constraints on the sensing range for both minimal and maximal distance. The left plot corresponds to the holonomic case while the right plot presents the results when non-holonomic constraints are considered. These are a first order differentiability constraint on the height and a minimum turning radius of 0.25 m in the xy plane. The path length is 9.4 m and 13.1 m for the holonomic and non-holonomic cases respectively. (a) Holonomic case. (b) Non-holonomic case.

recorded MAV paths. Regarding the computation time, first feasible solutions are found within a few minutes, while finding a well optimized path takes in general significantly longer.

The presented inspection paths ensure full coverage of the desired 3D structure. This theoretical result was experimentally evaluated and validated by using the feeds of the onboard vision system and executing 3D reconstruction algorithms. Two reconstruction strategies were employed, namely (a) that of the free online tool 123D Catch[24] which uses images from only one of the onboard cameras to conduct the reconstruction process using computer vision algorithms, and (b) a stereo dense reconstruction pipeline that uses OctoMaps to represent the environment.[25] The results are shown in Fig. 8 and as can be seen complete reconstruction was achieved for both holonomic and non-holonomic cases. Note that a recorded response of the last experiment can be found in the video provided following this link: `https://youtu.be/e7ljyDM9h8o`.

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

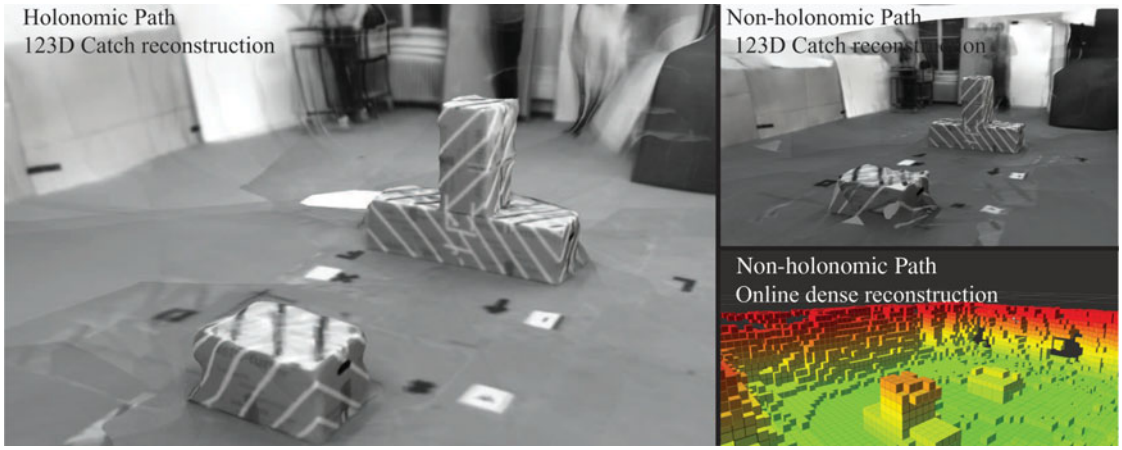


Fig. 8. 3D reconstruction results for the holonomic and non-holonomic paths using the offline 123D Catch tool as well as an online dense reconstruction pipeline that combines stereo block matching techniques fused with position information to derive a voxel-based representation. As shown, satisfactory reconstruction results are derived for both types of paths.

6. Summary & Conclusions

A new sampling-based inspection planning algorithm was presented within this work. The algorithm, the performance of which relies on a rewiring mechanism on the vertices of its tree structure, was described, followed by analysis on its capability to find admissible full coverage solutions with decreasing cost. The proposed RRTOT planner was evaluated in both simulation and experimental cases. Good inspection paths were found considering both, a holonomic and a particular case of non-holonomic vehicle with a turn radius limitation. The experimental case studies were conducted employing a hexacopter miniature aerial vehicle and an onboard vision module used for inspection. The derived experimental results and the corresponding 3D reconstructions of the inspected structures indicate the high quality of the computed solutions. Future work includes adaptations and experiments to plan for fixed-wing aircrafts as well as finding further heuristic concepts to speed up the computation while maintaining the theoretical optimality guarantees. Furthermore, one could consider employing the basic steps that are executed at each iteration of this algorithm and run them in an online fashion with the aim to iteratively explore unmapped areas (using real-time sensor data). In such a case, the goal would not be to guarantee global optimality but rather a locally optimized path to maximize the expected information gain.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Grant Agreement No.644128, AEROWORKS.

References

1. E. Stumm, A. Breitenmoser, F. Pomerleau, C. Pradalier and R. Siegwart, "Tensor-voting-based navigation for robotic inspection of 3d surfaces using lidar point clouds," **31**(12), 1465–1488 (2012).
2. M. Burri, J. Nikolic, C. Hurzeler, G. Caprari and R. Siegwart, "Aerial Service Robots for Visual Inspection of Thermal Power Plant Boiler Systems," *Applied Robotics for the Power Industry, 2012 2nd International Conference on* Zurich, Switzerland (2012) pp. 70–75.
3. B. Englot and F. S. Hover, "Three-dimensional coverage planning for an underwater inspection robot," **32**(9–10), 1048–1073 (2013).
4. A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel and R. Siegwart, "Structural Inspection Path Planning Via Iterative Viewpoint Resampling with Application to Aerial Robotics," *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA (May 2015) pp. 6423–6430. [Online]. Available: `https://github.com/ethz-asl/StructuralInspectionPlanner`
5. A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel and R. Siegwart, "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," *Autonomous Robots*, Springer US, 1–25 (2015). DOI: 10.1007/s10514-015-9517-1, ISSN: 1573–7527.

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press

6. E. Galceran and M. Carreras, Marc, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.* **61**(12), 1258–1276 (2013).
7. J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Robot. Res.* **10**, 628–649 (1991).
8. H. Choset, "Coverage for robotics–a survey of recent results," *Ann. Math. Artif. Intell.* **31**(1–4), 113–126 (2001).
9. A. Zelinsky, R. A. Jarvis, J. Byrne and S. Yuta, "Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot," *Proceedings of International Conference on Advanced Robotics* vol. 13, Tsukuba (1993) pp. 533–538.
10. J. Urrutia, "Art Gallery and Illumination Problems," **In**: *Handbook of Computational Geometry*, Instituto de Mathematicas, Universidad Nacional Autonoma de Mexico: Mexico (2000) pp. 973–1027.
11. D. Shmoys, J. Lenstra, A. Kan and E. Lawler, *The Traveling Salesman Problem*. J. K. Lenstra, A. R. Kan, E. L. Lawler, D. B. Shmoys, editors. The traveling salesman problem: a guided tour of combinatorial optimization. John Wiley & Sons (1985).
12. B. Englot and F. S. Hover, "Sampling-Based Sweep Planning to Exploit Local Planarity in the Inspection of Complex 3d Structures," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Vilamoura (2012) pp. 4456–4463.
13. T. Danner and L. E. Kavraki, "Randomized planning for short inspection paths," *IEEE International Conference in Robotics and Automation*, 2000, Apr 24, Vol. 2, pp. 971–976, San Francisco, CA, USA.
14. P. S. Blaer and P. K. Allen, "View planning and automated data acquisition for three-dimensional modeling of complex sites," *J. Field Robot.* **26**(11–12), 865–891.
15. G. Papadopoulos, H. Kurniawati and N. Patrikalakis, "Asymptotically Optimal Inspection Planning using Systems with Differential Constraints," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany (2013) pp. 4126–4133.
16. S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.* **30**(7), 846–894 (2011).
17. E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd and L. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," *IEEE Trans. Robot.* **21**(4), 597–608 (2005).
18. W. Wang, L. Yan, X. Xu and S. X. Yang, "An Adaptive Roadmap Guided Multi-RRTs Strategy for Single Query Path Planning," *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA),* IEEE, Anchorage, AK, USA (2010) pp. 2871–2876.
19. D. Devaurs, T. Simeon, J. Cortés *et al.*, "A Multi-Tree Extension of the Transition-Based RRT: Application to Ordering-and-Pathfinding Problems in Continuous Cost Spaces," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA (2014) pp. 2991–2996.
20. D. Hsu, R. Kindel, J. C. Latombe and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, **21**(3), 233–255 (2002 March). doi: 10.1177/027836402320556421.
21. M. W. Achtelik, S. Lynen, M. Chli and R. Siegwart, "Inversion Based Direct Position Control and Trajectory Following for Micro Aerial Vehicles," *IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, Tokyo, JP (2013) pp. 2933–2939.
22. J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale and R. Siegwart, "A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time Slam," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* IEEE, Hong Kong (2014) pp. 431–437.
23. Ascending Technologies GmbH, "http://www.asctec.de/".
24. Autodesk Inc., "http://www.123dapp.com/catch".
25. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots* **34**(3), 189–206 (2013).

https://doi.org/10.1017/S0263574716000084 Published online by Cambridge University Press