Cooperative coverage path planning for visual inspection

Article *in* Control Engineering Practice · March 2018

DOI: 10.1016/j.conengprac.2018.03.002

5 authors, including:

Sina Sharif Mansouri
Luleå University of Technology
23 PUBLICATIONS **87** CITATIONS
SEE PROFILE

Dariusz Kominiak
Luleå University of Technology
13 PUBLICATIONS **55** CITATIONS
SEE PROFILE

George Nikolakopoulos
Luleå University of Technology
221 PUBLICATIONS **2,402** CITATIONS
SEE PROFILE

Cooperative Coverage Path Planning for Visual Inspection ☆

Sina Sharif Mansouri[a], Christoforos Kanellakis[a], Emil Fresk[a], Dariusz Kominiak[a], George Nikolakopoulos[a]

[a]*Robotics Group, Control Engineering Division of the Department of Computer, Electrical and Space Engineering, Luleå University of Technology, Luleå SE-97187, Sweden. Emails:* {sinsha,chrkan,emil.fresk,darkom,geonik}@ltu.se

Abstract

This article addresses the inspection problem of a complex 3D infrastructure using multiple Unmanned Aerial Vehicles (UAVs). The main novelty of the proposed scheme stems from the establishment of a theoretical framework capable of providing a path for accomplishing a full coverage of the infrastructure, without any further simplifications (number of considered representation points), by slicing it by horizontal planes to identify branches and assign specific areas to each agent as a solution to an overall optimization problem. Furthermore, the image streams collected during the coverage task are post-processed using Structure from Motion, stereo SLAM and mesh reconstruction algorithms, while the resulting 3D mesh can be used for further visual inspection purposes. The performance of the proposed Collaborative-Coverage Path Planning (C-CPP) has been experimentally evaluated in multiple indoor and realistic outdoor infrastructure inspection experiments and, as such, it is also contributing significantly towards real life applications for UAVs.

Keywords: Cooperative Coverage, Path Planning, Visual Inspection, UAVs, Application

☆This work has received funding from the EU Horizon 2020 Research and Innovation Programme under the Grant Agreement No.644128, AEROWORKS.

1. Introduction

The annual investments in the infrastructure sector represent a significant percentage of the Gross Domestic Product (GDP) of the developed and developing countries, e.g. 3.9% of the GDP for the old European states, 5.07% of the GDP for the new European states and 9% of the GDP for China [1]. In order to reduce the risks to human lives and to increase the performance of the overall inspection procedure, autonomous ground, aerial or maritime vehicles are employed for executing inspection tasks. A few examples of applications are the following: power-line monitoring using autonomous mobile robots [2], bridge inspection [3], boiler power-plant 3D reconstruction [4], urban structure coverage [5], forest fire inspection using UAVs [6], and the inspection of underwater structures or ship hulls as in [7] and [8] respectively by the utilization of autonomous underwater vehicles. In most of these scenarios, there is an a priori knowledge about the infrastructure, where the available 2D-3D models can be derived by utilizing classical CAD software, sensor based reconstruction missions, civil engineering instrumentation and Geographical Information System (GIS) data.

In general, the task of Coverage Path Planning (CPP) [9] has received significant attention over the last years. However, there are very few CPP approaches in the field of aerial robotics; especially when CPP concept is extended to the Collaborative approach (C-CPP). In C-CCP, the utilization of multiple aerial agents has the potential to dramatically reduce the overall coverage time considering the flying times of all UAVs [10] and the levels of autonomicity. Thus, inspired by this vision, the main objective of this article is to establish a C-CPP method that is based on a priori knowledge of the infrastructure (e.g. a CAD model) and will have the ability to generate proper way points by considering multiple agents, while ensuring full coverage and overall collision avoidance among the flying agents. The proposed novel scheme creates a sub-coverage path planning for cooperative inspection of the whole infrastructure, while having the capability to detect branches of complicated infrastructures, which can

be assigned to different agents. Additionally to the coverage task, monocular or stereo based 3D reconstruction routines will be applied to build a 3D mesh of the structure. The image data captured during the task are processed using as basis existing technologies to generate sparse dense point clouds and 3D detailed models, combined in global representation areas covered by individual agents. In this manner the inspection process is automated to a great extent, demonstrating its real life applicability.

In the related literature there have been many works that addressed the CPP problem in 2D spaces [11] and fewer approaches that addressed coverage of 3D spaces. In [9], a complete survey was presented on CPP methods in 2D and 3D. Towards the 3D CPP, Atkar et al. [12] presented an off-line 3D CPP method for the spray painting of automotive parts. Their method used a CAD model and the resulting CPP could satisfy certain requirements for paint decomposition. In [13], the authors presented a CPP with real time re-planning for inspection of 3D underwater structures, where the planning assumed a priori knowledge of a bathymetric map using an autonomous underwater vehicle, while their overall approach contained no branches. The authors in [8] introduced a new algorithm for producing paths that cover complex 3D environments. The algorithm was based on off-line sampling for autonomous ship hull inspection, while the presented algorithm was able to generate paths for structures with unprecedented complexity.

In the area of aerial inspection, [5] presented a time-optimal UAV trajectory planning for 3D urban structure coverage. In this approach, the structures to be covered (buildings) were initially simplified into hemispheres and cylinders and in a later stage the trajectories were planned to cover these simple surfaces. In [14], the authors studied the problem of 3D CPP via viewpoint re-sampling and tour optimization for aerial robots. More specifically, they presented a method that supports the integration of multiple sensors with different fields of view and considered the motion constraints on aerial robots. Moreover, in the area of multi-robot coverage for aerial robotics in [15], a coverage algorithm with multiple UAVs for remote sensing in agriculture was proposed, where the target

area was partitioned into $k \in \mathbb{N}$ non-overlapping sub-tasks and in order to avoid collision, both different altitudes were assigned to each UAV and security zones were defined, where the vehicles were not allowed to enter.

Based on the aforementioned state of the art, the main contribution of this article is two-fold. The first significant contribution stems from the establishment of a theoretical framework, capable of providing a path for accomplishing a full coverage of the infrastructure, without a further shape simplification (number of considered representation points), by slicing it by horizontal planes to identify branches and assign specific areas to each agent, as a solution to an overall optimization problem, while the overall scheme is being presented for the case of multiple UAVs. This contribution comes in contrast to many existing approaches that simplify the infrastructure to an area of interest and solve it by various optimization methods; methods that could not be applied otherwise due to the inherent NP-hard complexity of the problem [16]. More specifically, the method presented in [17] has been implemented for only one agent and does not consider branches of the structure, an issue that have been considered in the proposed approach in this article. In the proposed framework, the a priori coverage path is divided and assigned to each agent based on the infrastructure architectural characteristics. Furthermore, to guarantee a full coverage and 3D reconstruction, the introduced path planning for each agent creates an overlapping visual inspection area that will enable the off-line cooperative reconstruction. In the novel established C-CPP scheme, the introduced algorithm takes into account the non-convex structure and identifies its branches. The algorithm, additionally to the position references, provides yaw references for each agent to assure a field of view, directed towards the structure surface.

The second major contribution stems from the direct demonstration of the applicability and feasibility of the overall novel C-CPP scheme for both indoors (simple structure) and outdoors (complex structure) by multiple aerial agents for reducing the inspection time in extended experimental trials. This demonstration has significant novelty and impact as an enabler for a continuation of research efforts towards the real-life aerial cooperative inspection of aging in-

frastructure, a concept that has never been presented before, to the authors best knowledge, outdoors and with a real infrastructure as a test case.

In the outdoor demonstrations, a case depicted in Figure 1 is considered, where the collaborative inspection UAVs have been autonomously operated based on odometry information from visual and inertial sensor fusion and without any other support on localization (e.g. motion capturing system), which adds complexity and highlights the impact of the acquired results. In this real-

Figure 1: Hexacopter during outdoors coverage task.

life application scenario, the image and pose data on board the UAV were post processed to build a 3D representation of the structure.

The rest of the article is structured as follows. The proposed C-CPP method is presented in Section 2, which is followed by a brief description of the 3D reconstruction part from multiple agents in Section 3. In Section 4 multiple simulation and experimental results are presented that prove the efficiency of the established scheme. Finally, the article is concluded in Section 5.

2. Coverage Path Planning of 3D Maps

For the establishment of the C-CPP, initially the general case of a robot equipped with a limited Field of View (FOV) sensor was considered, determined by an aperture angle α and a maximum range r_{max}, as depicted in Figure 2. Furthermore, $\Omega \in \mathbb{R}^+$ is the user-defined offset distance ($\Omega < r_{max}$), from the

infrastructure's target surface and $\Delta\lambda$ is the distance between each slice plane. $\Delta\lambda$ is equal to $\frac{\Omega}{2}\tan\alpha/2$ based on Figure 2. In order to guarantee overlapping, the parameter $\beta \in [1, +\infty)$ was introduced and substituted in $\Delta\lambda$ formulation that resulted to $\Delta\lambda = \frac{\Omega}{\beta}\tan\alpha/2$, where β represents the ratio of overlapping. This means that in the case where $\beta = 1$, it results in a minimum overlapping and when $\beta \to +\infty$, it reaches the maximum overlapping between each slice.

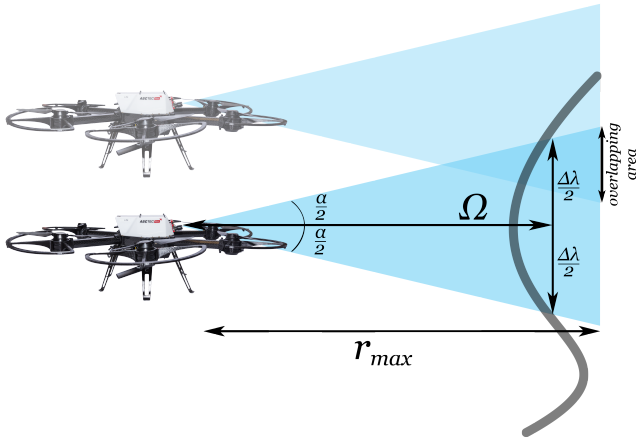


Figure 2: UAV based FOV (regenerated from [7]).

The proposed C-CPP method operates off line and it assumes that the 3D shape of the infrastructure is known a priori. The execution of the proposed cooperative inspection scheme is characterized by the following six steps: 1) slicing and intersection, 2) clustering, 3) adding offset from the infrastructure's surface, 4) path assignment for each UAV, 5) trajectory generation, and 6) online collision avoidance. This algorithmic approach is depicted in Figure 3. It should be noted that this scheme has extended the approaches presented in [12] and [17] to multiple robots. Additionally, it should be highlighted that the main scope of the paper is not to find the optimal path but rather to solve the problem of cooperative coverage under geometric approaches. This problem could be formulated as a Mixed-Integer nonlinear programming one that has been already studied in the literature and it is well known as the Multiple

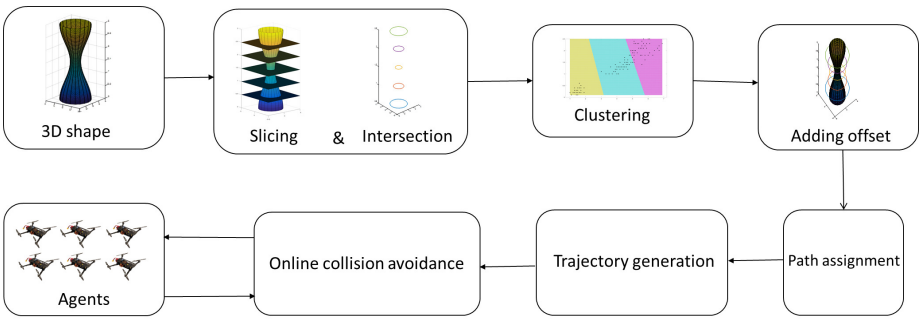


Figure 3: Block Diagram of the overall proposed C-CPP scheme.

Traveling Salesman problem. This is an Non-deterministic Polynomial-time hardness (NP-hard) problem, where the complexity is exponentially increased by the number of cities/points. Thus, solving the complete problem is non-feasible and is not studied in this article.

2.1. Slicing and Intersection

The 3D map of the infrastructure is provided as a set $\boldsymbol{S}$ with a finite collection of points, denoted as $\boldsymbol{S} = \{p_i\}$, where $i \in \mathbb{N}$ and $p_i = [x_i, y_i, z_i]^\top \in \mathbb{R}^3$. To be more specific, $\boldsymbol{S}$ represents a point cloud of the 3D object under inspection, where the points p_i, $i \in \mathbb{N}$, are defined by x, y and z coordinates and can be extracted from CAD or a 3D model of the structure. Additionally, in the concept of aerial inspection and based on all the use case scenarios that have been examined, there is always a practical way in approaching an infrastructure with a simpler geometrical 3D shape. This assumption can be applied to almost all man-made structures without loss of generality, while the dimensions of this 3D shape simplistic approach can be realized based on in situ simple measurements or by blueprints. As it will be presented in the reconstruction results, the initial assumption of the 3D shapes is needed only for the determination of the path planning, while the final reconstructed results and point clouds have the required quality for creating a detailed and dense reconstruction. Thus, in the field of aerial inspection, even in the case that a detailed 3D model of the structure is provided, the structure model can be further simplified in order to

assist the generation of the flight paths for the UAVs.

As an illustrative example, in the complicated case of a wind turbine, the structure could be modeled by five different cylinder shapes for the tower, the hub and the three blades. This geometrical assumption does not reduce the generality of the geometric approach; at the same time it can be indirectly utilized as a means to create flying paths with an inherent level of safety for avoiding collisions with the infrastructure.

The map $\boldsymbol{S}$ is then sliced by multiple horizontal planes, defined as λ_i, with $i \in \mathbb{N}$. The value of λ_i started from $\min_z \boldsymbol{S}(x, y, z) + \Delta\lambda$ and ended with $max_z\boldsymbol{S}(x, y, z) - \Delta\lambda$, thus in this case, the horizontal plane translates vertically along the z-axis, while increasing the distance from the current slice, until reaching the maximum value. The intersection between the 3D shape and the slice can be calculated as follows:

$$\boldsymbol{\Sigma}_i = \{(x, y, z) \in \mathbb{R}^3 :< \vec{n}, \boldsymbol{S}(x, y, z) > -\lambda_i = 0\} \tag{1}$$

where $\boldsymbol{\Sigma}_i(x, y, z)$ are the points of the intersection of the plane, $\boldsymbol{S}$, $\vec{n} \in \mathbb{R}^3$ is the vector perpendicular to the plane, λ_i defines the location of the plane and $< ., . >$ is the inner product operation. In the examined case, for simplicity the planes are chosen to be horizontal $\vec{n} = [0, 0, 1]^\top$. The slicing and intersection scheme is presented in Algorithm 1, while the overall concept is depicted in Figure 4.

Algorithm 1 Slicing and Intersection

Require: Points of 3D shape $\boldsymbol{S}(x, y, z)$, $\Delta\lambda$

1: $\lambda = min_z\boldsymbol{S}(x, y, z) + \Delta\lambda$
2: **while** $\lambda_i \leq max_z\boldsymbol{S}(x, y, z)$ **do**
3: $\quad\boldsymbol{\Sigma}_i = \{(x, y, z) \in \mathbb{R}^3 :< \vec{n}, \boldsymbol{S}(x, y, z) > -\lambda_i = 0\}$
4: $\quad$Clustering() ; see Section 2.2.2
5: $\quad\lambda_{i+1} = \lambda_i + \Delta\lambda$
6: **end while**

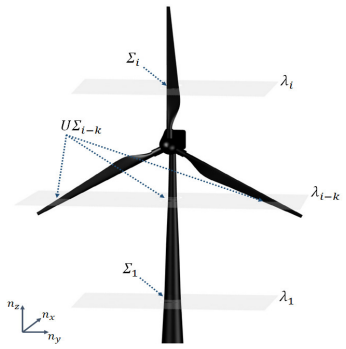


Figure 4: Concept of plane slicing and intersection points.

2.2. Clustering

In the case of convex infrastructures, there is at most one cluster of points in any slice, while for complex structures there will be more than one clusters in each slice. A cluster is defined by the number of branches of the structure e.g. Figure 5 shows the intersection points $\mathbf{\Sigma_i}(x, y, z)$ of the non-convex object with multiple branches in 2D which results in a maximum of 3 clusters in λ_{i-k}. The set of points in $\mathbf{\Sigma}_i(x, y, z)$ (as obtained from Algorithm 1) is multi-modal. Thus, in order to complete the overall C-CPP task, it is critical to recognize the number of clusters and group the data elements (via clustering) for further task assignments of the agents.

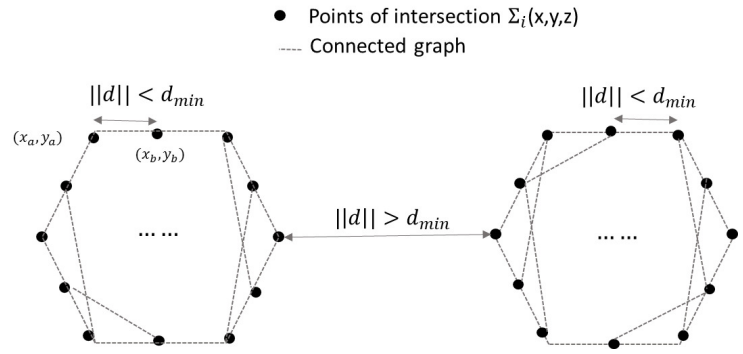


Figure 5: Intersection points of non-convex object with multiple branches.

2.2.1. Number of clusters

Clusters in $\mathbf{\Sigma}_i(x, y, z)$ are defined through connectivity by calculating the number of paths that exist between each pair of points. For some points to belong to the same cluster, they should be highly connected to each other. As shown in Figure 5 they can be connected based on their distances. In order to obtain the number of clusters, graph theory [18] will be utilized, thus initially the adjacency matrix A is generated. The adjacency matrix A is formed based on the euclidean distance $||d||$ of the points. If the distance of two points is less than d_{min}, then the points are connected. Moreover, for selecting the value of d_{min}, the size of the agent is considered; if the agent cannot pass through two points, they are assumed to be connected. In the next step, the degree matrix $D(a, b)$ is generated, which is the diagonal matrix and the elements of the diagonal $D(a, a)$ show the number of connected points to the a^{th} point. Later on, the Laplacian matrix L is calculated from the subtraction of the adjacency matrix and the degree matrix $A - D$. Finally, the eigenvalues of L are calculated. The number of zero eigenvalue corresponds to the number of the connected graphs. The detailed process is presented in Algorithm 2.

2.2.2. Graph Clustering

The existing global clustering approaches are capable of dealing with up to a few million points on sparse graphs [19]. More specifically, in global clustering, each point of the input graph is assigned to a cluster in the output of the method. Mixed Integer Programming (MIP) clustering and K-means clustering approaches can be used as a global method for graph clustering.

In general, the *K-means* methods objective function can be solved in different approaches, while in this article the minimization of the point's distance to the center of the k clusters is considered. This formulation can be solved by different optimization algorithms, such as sequential quadratic programming [20] or MIP [21]. An alternative approach to this problem is to formulate the optimization problem as a minimization of the diameter of the clusters and solve it by MIP [22] or Brunch-and-Bound algorithms [23, 24]. Each method has its own

Algorithm 2 Number of clusters

Require: $\mathbf{\Sigma}_i(x, y, z)$, d_{min}

1:

2: **for** a , $b \in \mathbf{\Sigma_i}$ **do** ;a,b are two pairs of points

3: $||d|| = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$

4:

5: **if** $d < d_{min}$ **then**

6: $A(a, b) = 1$

7: **end if**

8: **end for**

9: $D(a, b) = \begin{cases} \#\text{ of ones in } a^{th}\text{row of } A & a == b \\ 0 & a \neq b \end{cases}$

10: $L = A - D$; Laplacian matrix

11: $E = eig(L)$; eigen values of Laplacian matrix

12: $k = \#\, of\, E == 0$; Number of clusters

pros and cons that have been presented extensively in the relevant literature, e.g. in [19] where it has been presented that explicit optimization techniques such as MIP or Branch and bound (BnB) algorithms can be used for reducing the necessary enumerations, but even with such savings, global searches remain impractical due to the fact that they are computationally expensive and complete enumeration of every possible partition is not possible [25]. On the other hand, *K-means* algorithm can scan a large data set once and produces a clustering using small memory buffers and thus it has been selected in the presented approach. In general the clustering approach is a unique problem itself and it will not be considered in details in this article, however *K-means* and MIP diameter clustering methods are studied and compared.

The challenge in using clustering algorithms is to determine the number of the clusters in a data set (Algorithm 2). The *K-means* clustering algorithm is used with a priori knowledge of the number of clusters k. The *K-means* clustering [26] is a combinatorial optimization problem that assigns m sets to

exactly one of k ($k < m$) clusters, while minimizing the sum of the distances of each point in the cluster to its respective center. The utilization of the cluster algorithm can categorize the points, independently of the order of them or the direction of slicing to the object, while providing the center of each cluster, which is utilized for the calculation of the reference yaw angle for the agents. The algorithm is presented in 3.

Algorithm 3 K-means

Require: $\boldsymbol{\Sigma_i}(x, y, z)$, k

1: Choose k initial cluster centers C_i.
2: Calculate $\min\limits_{\boldsymbol{\Sigma_i}} \sum\limits_{i=1}^{k} \sum\limits_{\boldsymbol{\Sigma_i}} ||[x, y, z]^\top - C_i||$
3: Assign points to the closest cluster center.
4: Obtain k_{new} centers by computing the average of the points in each cluster
5: Repeat lines 2 to 4 until cluster assignments do not change

Moreover, in the MIP diameter clustering method, the objective function is to minimize the maximum diameter of the generated clusters with the goal of obtaining compact clusters. This criterion was previously studied in [22]. It is assumed that the number of desired k clusters is known from Section 2.2.1, while the mathematical formulation of the model MIP diameter, is provided in the following Equation (2a).

$$\min\ D_{max} \tag{2a}$$

$$s.t.\ D_c > d_{ij} x_{ic} x_{jc}\ \forall i, j = 1, \ldots, n,\ c = 1, \ldots, k \tag{2b}$$

$$\sum_{c=1}^{c=k} x_{ic} = 1 \tag{2c}$$

$$D_{max} \geq D_c\ \forall c = 1, ..., k \tag{2d}$$

$$x_{ic} \in \{0, 1\}\ \forall i = 1, \ldots, n,\ c = 1, \ldots, k \tag{2e}$$

$$D_c \geq 0\ \forall c = 1, \ldots, k \tag{2f}$$

where D_c is the diameter of the cluster c, D_{max} is the maximum diameter among the generated clusters, d_{ij} is the distance between two points, n is the

total number of points, x_{ic} is the binary decision variable that represents the assignment to a particular cluster, and equals 1 if i is assigned to c and 0 otherwise. Equation (2b) ensures that the diameter of cluster c is allowed to be at least the maximum distance between any two data points in cluster c, while Equation (2c) guarantees that each point is assigned to only one cluster and Equation (2d) sets the variable D_{max} equal to the value of the maximum diameter. The optimization has kn binary variables and $k+1$ continuous variables. As it is suggested in [22], a MIP CPLEX solver [27] and a heuristic approach is used to solve the problem. In Figure 6, a dataset is presented where the global optimal solution for a clustering is known a priori. In each simulation a dataset with different number of points n is studied and the corresponding performance of the MIP clustering and *K-means* is compared. All the simulations have been performed in MATLAB on a computer with an Intel Core i7-6600U CPU, 2.6GHz and 8GB RAM.

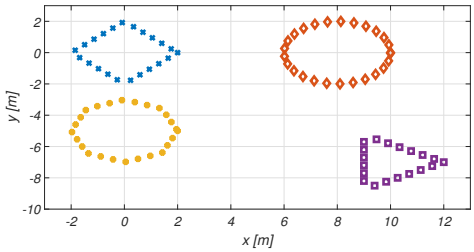


Figure 6: Dataset for 4 clusters.

Furthermore, Figure 7 shows that the computational time of the MIP clustering method is increasing exponentially with the number of points when compared to the *K-means* algorithm. Thus, in this article, the *K-means* has been utilized, without loss of generality with an overall aim for an online application of the proposed overall scheme and additionally, a case by case comparison is presented in Section 4 with a corresponding discussion.

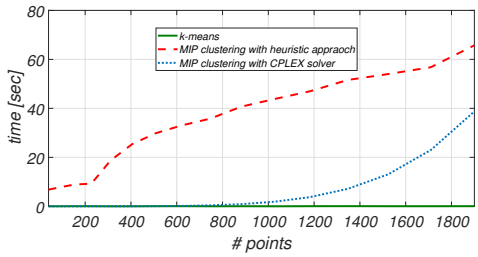


Figure 7: Computation time of the MIP clustering method and the *K-means* compared to the number of the points.

2.2.3. Convex Hull

It is important for the coverage algorithm that waypoints are not inside the object and guarantee full surface inspection. In the case that the provided 3D model is not hollow, some waypoints are located inside the object. These points are not reachable during coverage mission and should not be considered. Therefore, a convex hull [28] algorithm is used to form the smallest convex set that contains the points in $\mathbb{R}^2$. However, using the convex hull, without clustering the point sets, results in a loss of some parts of the object in $\mathbb{R}^2$. Figure 8 depicts the result of the convex hull before the generation of clustered point sets. In order to provide convex sets without loss of generality the *quickhull* [28] algorithm is used 4.

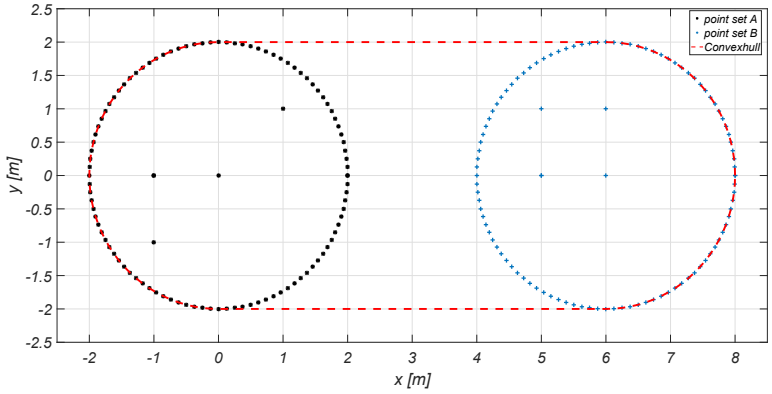


Figure 8: Convex hull example without clustering.

Algorithm 4 *Quickhull* for the convex hull in $\mathbb{R}^2$

Require: the set $Cluster_i$ of n points

function QUICKHULL

Find the points with minimum and maximum x coordinates $p_i = \{(x_{min}, y_i)\}$, $p_j = \{(x_{max}, y_j)\}$.

Segment $p_i p_j$ divides the remaining $(n-2)$ points into 2 groups G_i and G_j

convexhull=$\{p_i, p_j\}$

FINDHULL(G_i, p_i, p_j)

FINDHULL(G_j, p_j, p_i)

end function

function FINDHULL(G_k, p_i, p_j)

if $G_k = \{\}$ **then**

Return

end if

Find the farthest point p_f, from segment $p_i p_j$

Add point p_f to the convex hull at the location between p_i and p_j

$Cluster_i$ is divided into three subspaces by p_i, p_j, and p_f.

convexhull=convexhull $\cup \{p_f\}$

FINDHULL(G_i, p_i, p_f)

FINDHULL(G_j, p_f, p_j)

end function

2.3. Adding Offset

In the proposed C-CPP scheme, the UAV should cover the offset surface, which has a fixed distance Ω from the target surface, with the size of the agent also considered in Ω. A distance that can be considered as a safety distance Ω is kept constant and it is continuously adapted to the infrastructures characteristics. For a considered set of cluster $\mathcal{C}(x, y, z)$, its offset path OP can be written as

$$OP(x_j, y_j, z_j) = \mathcal{C}(x_j, y_j, z_j) \pm \Omega \vec{n}_j, \tag{3}$$

where $\vec{n}_j$ is the normal vector at the j^{th} point and $\pm$ presents the direction of the normal vector; more details about the addition of the offset strategy can be found in [29]. Figure 9 shows some examples in calculating OP for four different shapes of infrastructure, such as circular, square, hexagonal and triangular. In each of these shapes, the desired offset path with $\Omega = 1\,\text{m}$ was calculated, while in all the cases, the UAV flew parallel to the edges, with a constant distance of Ω until the next vertex was reached and while keeping the Ω distance while flying to the next edge.

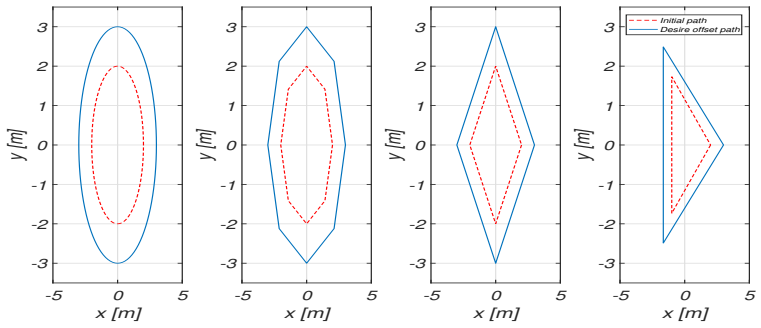


Figure 9: Example of adding offset to different shapes.

2.4. Path Assignment

For sharing the aerial infrastructure inspection to multiple UAVs, two different cases have been considered; one where there is only one ($m = 1$) branch, and the other where there are multiple branches ($m > 1$). In the first case,

each agent should cover a part of the branch, while in the case of two agents θ_a and θ_b have a difference of 180° for equal space between the points. In the case of more than one branch, the UAVs should be equally distributed between the branches, where the policy of assigning each agent to each branch is shown in Algorithm 5.

Algorithm 5 C-CPP based assignment of UAVs to branches policy

1: Assume to have n agents and m branches ($n \geq m$).
2: **if** $m == 1$ **then**
3: All n agents go in the same slice.
4: **end if**
5: **if** $m > 1$ AND $n > 1$ **then**
6: n agent equally distributed.
7: **end if**

2.5. Trajectory Generation

The resulting waypoints are then converted into position-velocity-yaw trajectories, which can be directly provided to the utilized linear model predictive controller, cascaded [30] over an attitude-thrust controller. This is done by taking into account the position controller's sampling time T_s and the desired velocity along the path $\vec{V}_d$. These trajectory points are obtained by linear interpolation between the waypoints, in such a way that the distance between two consecutive trajectory points equals the step size $h = T_s||\vec{V}_d||$. The velocities are then set parallel to each waypoint segment and the yaw angles are also linearly interpolated with respect to the position within the segment. The adopted trajectory generation that was used in the experimental realization of the proposed C-CPP is depicted in Figure 10 with $||\vec{V}_d|| = 0.5\text{m/s}$ and $T_s = 1\,\text{s}$.

2.6. Collision Avoidance

In order to avoid collisions, the following method is used to guarantee the maximum distance between the UAVs. Assuming n agents and a set of points

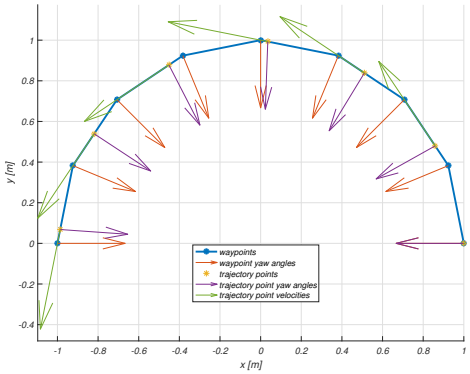


Figure 10: Example application of the proposed waypoint-to-trajectory conversion algorithm.

$\mathcal{P}_i(x, y, z) \subset \mathbf{\Sigma}_i(x, y, z)$ assigned for i^{th} agent, the following optimization problem is solved sequentially for each agent to find the points with a safety distance with respect to the other agents and the minimum distance to the current position of the agent (Equation (4)). The collision avoidance scheme that will be described later has been performed under the following assumptions: 1) the Ω is large enough to avoid the collision of agents to the object, 2) the distance between the branches of the object is more than the safety distance d_s so the agents in separate branches cannot collide, and 3) entering and leaving of the agents, for each cluster, is a collision free path.

$$\begin{aligned} \min_{p_i} \; & ||p_i - p_i^*|| \\ & ||\vec{d_j}|| > d_s \\ & p_i(x_j, y_j, z_j) \in \mathcal{P}_i \\ & j \geq 2 \end{aligned} \tag{4}$$

where p_i is the current position, p_i^* is the future position of the i^{th} agent, $||\vec{d_i}||$ contains the distances of the i^{th} agent from the rest of the agents.

The above optimization is an *Integer Linear Programming* problem, as the optimization procedure should find the $(x_j, y_j) \in \mathcal{P}_i$ for each agent in order to minimizes the distance of the waypoints, guaranteeing collision free paths for

each agent. The overall algorithm is also presented in 6, where the operator "\" denotes the relative complement calculation and is defined as follows:

$$\mathcal{P}_i \setminus p_i = \{(x, y, z) \in \mathcal{P}_i | \, (x, y, z) \notin p_i\} \tag{5}$$

Algorithm 6 Collision avoidance between agent.

Require: $p_i, \mathcal{P}_i(x, y, z), n, d_s$

1: **for** $i = 2, \ldots n$ **do**
2: Compute p_i^* using (4)
3: $p_i \leftarrow p_i^*$
4: $\mathcal{P}_i \leftarrow \mathcal{P}_i \setminus p_i$
5: **end for**

Moreover, when the agents have to change a branch, they may collide with the object as the C-CPP only provides the initial point (x_s, y_s, z_s) and the destination point (x_d, y_d, z_d) for moving the agent from one branch to another branch. Thus, in general, it is possible that the intermediate points in this path are closer than the safety distance or inside the object. For avoiding these cases, the line is produced by connecting the initial point and destination. If the points of the line $\mathcal{L}(x, y, z)$ are closer than the safety distance d_{so} to the object, an offset value d_{of} is added until the distance is larger than the safety distance (Algorithm 7). The direction of adding the safety distance corresponds to the center of the cluster as calculated in (Section 2.3).

3. Multiple agent Visual Inspection

As stated, the C-CPP method is targeting the case of autonomous cooperative inspection by multiple aerial UAVs. Each aerial platform is equipped with a camera to record image streams and provide a 3D reconstruction of the infrastructure. More specifically, two main approaches have been considered to obtain the 3D model of the infrastructure, using either stereo or monocular

Algorithm 7 Collision avoidance to the object.

Require: initial point=(x_s, y_s, z_s) and destination point=(x_d, y_d, z_d)

1: $\mathcal{L}(x, y, z) = \frac{x-x_s}{x_d-x_s} = \frac{y-y_s}{y_d-y_s} = \frac{z-z_s}{z_d-z_s}$
2: **for** $(x_a, y_a, z_a) \in \boldsymbol{\mathcal{L}}(x, y, z)$ **do**
3: $||d|| = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2}$
4: where $(x_b, y_b, z_b) \in \boldsymbol{S}(x, y, z)$
5: **if** $||d|| < d_{so}$ **then**
6: flag=1
7: **else**
8: Do NOTHING
9: **end if**
10: **end for**
11: **if** flag==1 **then**
12: Add offset value d_{of} to (x_s, y_s, z_s) and $(x_d, y_d, z_d) \rightarrow (x'_s, y'_s, z'_s)$ and (x'_d, y'_d, z'_d)
13: $\mathcal{L}(x, y, z) = \frac{x-x'_s}{x'_d-x'_s} = \frac{y-y'_s}{y'_d-y'_s} = \frac{z-z'_s}{z'_d-z'_s}$
14: go to line 2
15: **end if**

camera mapping. In both cases the aim is to merge the processed data from multiple agents into a global representation. The selection between these two approaches depends mainly on the application's needs and the object's structure. The perception of depth using stereo cameras is bounded by the stereo baseline, essentially reducing the configuration to monocular at far ranges. For the monocular case, the employed Structure from Motion (SfM) approach, that provides a 3D reconstruction and camera poses using different camera viewpoints, induces the need to solve a large optimization problem. In this article however, both methods were used as a proof of concept and will be presented in Section 4.

3.1. Stereo Mapping

Generally, in the stereo visual mapping case, each robot provides an estimate of its pose and a reconstruction of the environment in its local coordinate frame. In the developed case the agents localize themselves in the coordinate frame of the global localization system (e.g. motion capture system) and integrate this odometry information in the stereo SLAM [31] algorithm. All the generated pointclouds, from the individual agents, are expressed in the coordinate frame. In the case of n agents, each agent covers a specific area around the object of interest, reconstructing a unique map. Thereafter, each of the n created maps are merged through the Iterative Closest Point (ICP) [32] algorithm to form a global representation of the whole structure. This technique takes two point sets as inputs, $\boldsymbol{\Xi}$ and $\boldsymbol{Z}$ (Algorithm 8) and calculates the 3D transform τ that expresses the relative pose between them, using a set of filtered corresponding points $f(\boldsymbol{\Xi})$ and $f(\boldsymbol{Z})$. The transform is estimated by minimizing the error $E_{er} = \sum_{j=1}^{N} (\|\tau(\xi_j) - \zeta_i\|^2)$. Then, τ is applied to align the point sets in a common frame. The output is a global point set $\boldsymbol{M}$ that represents the 3D reconstruction of the whole scene. Thus, it is important that the agents cover a common part of the scene in order to align the individual maps. The process is performed off-line.

The basis for the 3D reconstruction is a state of the art stereo mapping

algorithm known as RTABMap [33] SLAM. This algorithm is suitable for large scale operations and fits well for complex structure mapping. More specifically, RTABMap is an appearance based Localization and Mapping algorithm that consists of three parts: the visual odometry, the loop closure detection, and the graph optimization part.

Initially, the algorithm identifies features in images and builds a visual word vocabulary [34] for loop closure processing. This process is used to determine if a new image location has been previously visited. Then, the next step is to project the feature locations from the image to 3D using the depth measurements, through a filtering step to remove outliers. If features are identified in previous frames, the frame is added as a node in the pose graph, while a Bayesian filter keeps track of the loop closure cases. When loop closure is identified, the image location is added in the graph that holds the map $\boldsymbol{M}_i$. Finally, pose graph optimization and Bundle Adjustment are performed to refine the estimated location. The aforementioned process is described in Algorithm 8.

3.2. Monocular Mapping

In the monocular mapping case, the incremental Structure from Motion (SfM) [35, 36] technique is used to build a reconstruction of the object under inspection. While the aerial agents follow their assigned path around the object of interest, the image streams from the monocular cameras of the agents are stored in a database. In the SfM process, different camera viewpoints are used off-line to reconstruct the 3D structure. The process starts with the correspondence search step, which identifies overlapping scene parts among input images. During this stage, feature extraction and algorithm matching between frames is performed to extract information about image scene coverage. Following this, the geometric verification using the epipolar geometry [37] to remove false matches takes place. In this approach, it is crucial to select an initial image pair $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ with enough parallax to perform two-view reconstruction, before incrementally registering new frames. Firstly, the algorithm recovers the sets of matched features f_1 and f_2 in both images. Next, it estimates the camera

Algorithm 8 Stereo 3D reconstruction

function ICP FRAME ALIGNMENT
Map 1 point set $\boldsymbol{\Xi} = \xi_1, \xi_2, \xi_3, \cdots, \xi_N$
Map 2 point set $\boldsymbol{Z} = \zeta_1, \zeta_2, \zeta_3, \cdots, \zeta_N$
Extract point features $(\boldsymbol{\Xi}) \rightarrow f(\boldsymbol{\Xi})$
Extract point features $(\boldsymbol{Z}) \rightarrow f(\boldsymbol{Z})$
Match $f(\boldsymbol{\Xi})$ and $f(\boldsymbol{Z})$
Filter false matches and remove outliers
$\min_{\tau} E_{er} = \sum_{j=1}^{n} (\|\tau(\xi_j) - \zeta_i\|^2)$
Apply transform τ and align point sets
end function
function VISUAL SLAM
Frame location intialization (Feature Detection, Visual word vocabulary)
Loop closure detection
Frame location to pose graph
Pose graph optimization
Pointcloud with corresponding pose $\rightarrow \boldsymbol{M}$
end function

Assume to have n agents

for agent i **do**
Visual SLAM $\rightarrow \boldsymbol{M}_i$; 3D map generated for each agent
end for
Merge Maps
for $i : 1 : \#maps$ **do**
ICP FRAME ALIGNMENT($\boldsymbol{M}_i$)$\rightarrow$ global map $\boldsymbol{M}$; Agents with common scene coverage
end for

extrinsics for $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ using the 5-point algorithm [38], decomposing the resulting Essential matrix $\boldsymbol{E_{es}}$ with Singular Value Decomposition (SVD) and finally builds the projection matrices $\boldsymbol{P}_i = [\boldsymbol{R}_i|\boldsymbol{t}_i]$ that contain the estimated rotation and translation for each frame. Then, using the relative pose information, the identified features are triangulated to recover their 3D position $\boldsymbol{X}^{3D}$. Afterwards, the two-frame Bundle Adjustment refines the initial set of 3D points, minimizing the reprojection error. After this initialization step, the remaining images are incrementally registered in the current camera and point sets. More specifically, the frames that capture the largest amount of the recovered 3D points are processed by the Perspective-n-Point (PnP)[39]. This algorithm uses 2D feature correspondences to 3D points to extract their pose. Furthermore, the newly registered images will extend the existing set of the 3D scene ($\boldsymbol{X}^{3D}$) using multi-view triangulation. Finally, a global Bundle Adjustment is performed on the entire model to correct drifts in the process. The aforementioned process is described in the algorithm 9. During the coverage tasks, the agents fly autonomously based on visual inertial odometry (Section 4). The absolute scale of the reconstructed object can be recovered by combining full-pose annotated images from the on-board localization of the camera.

4. Results

4.1. Experimental Setup

The proposed method has been evaluated with the utilization of the Ascending Technologies NEO hexacopter, depicted in Figure 11. The platform has a diameter of 0.59 m and height of 0.24 m. The length of each propeller is 0.28 m as depicted in Figure 11. This platform is capable of providing a flight time of 26 min, which can reach a maximun airspeed of 15 m/s and a maximum climb rate of 8 m/s, with maximum payload capacity up to 2 kg. It has an on-board Intel NUC computer with a Core i7-5557U and 8 GB of RAM. The NUC runs Ubuntu Server 14.04 with Robotic Operating System (ROS) installed. ROS is a collection of software libraries and tools used for developing robotic applica-

Algorithm 9 Monocular 3D reconstruction

function TWO-VIEW RECONSTRUCTION
Detect features (f_1, f_2) in frames ($\boldsymbol{I}_1$, $\boldsymbol{I}_2$)
Match f_1 and f_2 between $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$
Remove false matches $\rightarrow$ inlier matches ($\hat{f}_1$, $\hat{f}_2$) ; Geometric Verification
5-point($\boldsymbol{I}_1$, $\boldsymbol{I}_2$,f_1, f_2) $\rightarrow \boldsymbol{E}_s$; Essential Matrix
SVD($\boldsymbol{E}_s$) $\rightarrow \boldsymbol{R}$, $\boldsymbol{t}$; Relative Camera Pose
Projection matrices $\boldsymbol{P}_1$,$\boldsymbol{P}_2 \rightarrow \boldsymbol{P}_1 = [\boldsymbol{I}|\boldsymbol{0}]$, $\boldsymbol{P}_2 = [\boldsymbol{R}|\boldsymbol{t}]$ Triangulate($\boldsymbol{I}_1$, $\boldsymbol{I}_2$,f_1, f_2,$\boldsymbol{P}_1$,$\boldsymbol{P}_2$) $\rightarrow \boldsymbol{X}^{3D}$; 3D points
Bundle Adjustment(f_1, f_2,$\boldsymbol{P}_1$,$\boldsymbol{P}_2$,$\boldsymbol{X}^{3D}$)
end function
Import initial pair $\boldsymbol{I}_1$, $\boldsymbol{I}_2$
TWO-VIEW RECONSTRUCTION($\boldsymbol{I}_1$, $\boldsymbol{I}_2$,$\boldsymbol{X}^{3D}$)
for $i : 1 : \#frames$ **do**
Import new camera frame I_i
PnP($\boldsymbol{I}_i$,$\boldsymbol{X}^{3D}$) $\rightarrow \boldsymbol{P}_i$; i_{th} projection matrix, new frame registered
Multi-view Triangulation $\rightarrow \boldsymbol{X}^{3D}_{new} = \boldsymbol{X}^{3D}, \boldsymbol{X}^{3D}_{i,k}$; Augment existing 3D structure.
Bundle adjustment(f_i, $\boldsymbol{P}_i$,$\boldsymbol{X}^{3D}_{new}$)
end for

tions [40]. Additionally, multiple external sensory systems (e.g. cameras, laser scanners, etc.) can be operated in this setup. Regarding the on-board sensory

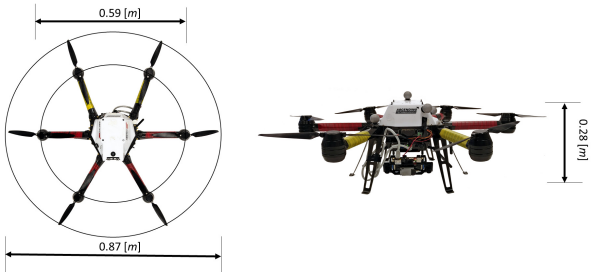


Figure 11: AscTec NEO platform with the VI sensor attached.

system, the Visual-Inertial (VI) sensor (weight of 0.117 kg Figure 11) developed by Skybotix AG is attached below the hexacopter with a 45 ° tilt from the horizontal plane. The VI sensor is a monochrome global shutter stereo camera with 78 ° FOV, housing an Inertial Measurement Unit (IMU). Both the cameras and the IMU [41] are tightly aligned and hardware synchronized. The camera was operated in 20 fps with a resolution of 752x480 pixels, while the depth range of the stereo camera lies between 0.4 and 6 m.

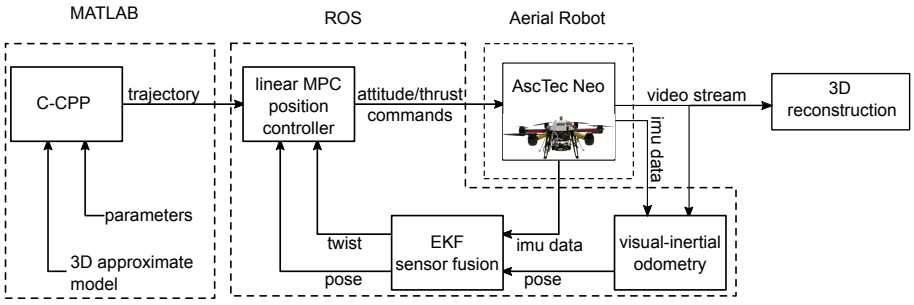


Figure 12: Software and hardware components used for conducting inspections.

The proposed C-CPP method, established in Section 2, has been entirely implemented in MATLAB. The inputs for the method are a 3D approximate model of the object of interest and specific parameters, which are the number

of agents n, the offset distance from the object Ω, the FOV of the camera α, the desired velocity of the aerial robot V_d and the position controller sampling time T_s. The generated paths are sent to the NEO platforms through the ROS framework.

The platform contains three main components to provide autonomous flight: a visual-inertial odometry, a Multi-Sensor-Fusion Extended Kalman Filter (MSF-EKF) [42] and a linear Model Predictive Control (MPC) position controller [43, 30, 44]. The visual-inertial odometry is based on the Robust Visual Inertial Odometry (Rovio) [45] algorithm for the pose estimation. It consists of an EKF filter that uses inertial measurements from the VI IMU (accelerometer and gyroscope) during the state propagation and the visual information is utilized during the filter correction step. The outcome of the visual inertial odometry is the position-orientation (pose) and the velocity (twist) of the aerial robot. Afterwards, the MSF-EKF component fuses the obtained pose information and the NEO IMU measurements. This consists of an error state Kalman filter, based on inertial odometry, performing sensor fusion as a generic software package, while it has the unique feature of being able to handle delayed and multi-rate measurements while staying within computational bounds. The linear MPC position controller [44] generates attitude and thrust references for the NEO predefined low level attitude controller. In Figure 13 the components of the controller are shown. Positions, velocity, and trajectory are sent to a Linear MPC position controller, which provides roll ϕ_d, pitch θ_d and mass normalized thrust T commands for the inner loop. From the initial control tuning experiments it was proven that the low level attitude controller is able to track the desired roll, pitch and yaw trajectory and to calculate the corresponding $n_1, n_2, \ldots, n_6$ rotor speeds for the vehicle. Details about the controller scheme and parameters' tuning can be found in [44].

The image stream from the overall experiment is processed using the method described in Section 3, while the overall schematic of the experimental setup is presented in Figure 12.

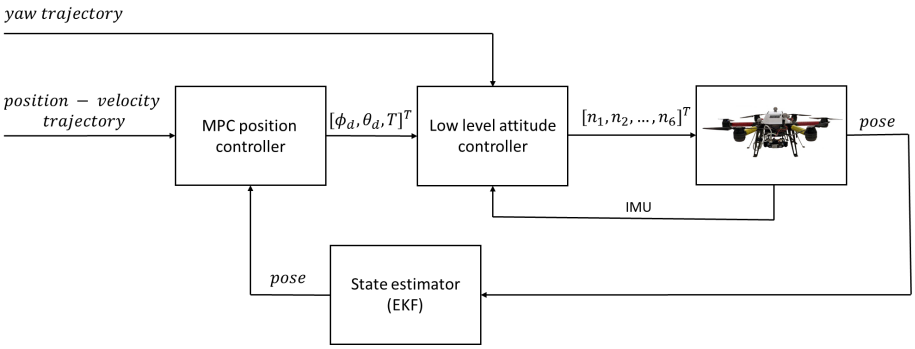


Figure 13: Controller scheme for NEO (regenerated from [44]).

4.2. Simulation and Experimental Evaluation

In this section, simulation and experimental results are presented to prove the concept of the proposed method. In the following cases the offset distance Ω from the inspection objects is 2 m in simulation, and 1 m and 3 m in indoor and outdoor experimental trials, respectively.

Initially, in order to evaluate the performance of the method, a wind turbine, a 3D complex structure with multiple branches, is selected for simulation purposes. The tower diameter is 4 m at the base, 1 m at the top and its height is 40 m. The blade length is 27 m with Cord length at the root of 2 m, cord length at the tip of 0.2 m and the volume of the hub and nacelle are $4 \times 5 \times 4\,\mathrm{m}^3$. In Figure 14 the paths, which are generated for one, two and three UAVs for the cooperative aerial inspection, are depicted. This structure has up to three branches as depicted in Figure 15, which can be recognized by the C-CPP. More specifically, the tower is considered to have one branch (green) until the height that two blades virtually intersect the horizontal plane of the tower. From that point, the structure is considered to have three branches (red), which are the two blades and the remaining part of the tower. Finally the nacelle and the third blade of the structure are considered as one branch (blue). In Figure (14 left), the whole structure is covered by one agent, while in the case of more than one agents (middle and right), the branches are assigned to each agent and in the case of a single branch, the area is shared between multiple agents. As a

result, the inspection time is significantly reduced as presented in Table 1.

Table 1: Inspection time for windmill inspection.

Number of agents	1	2	3
Inspection time [min]	24.86	17.63	11.36

Due to the lack of texture in the simulation, the quality of the reconstruction result from the recorded data is low and is therefore not included and more emphasis will be placed to the real-life experimental trials.

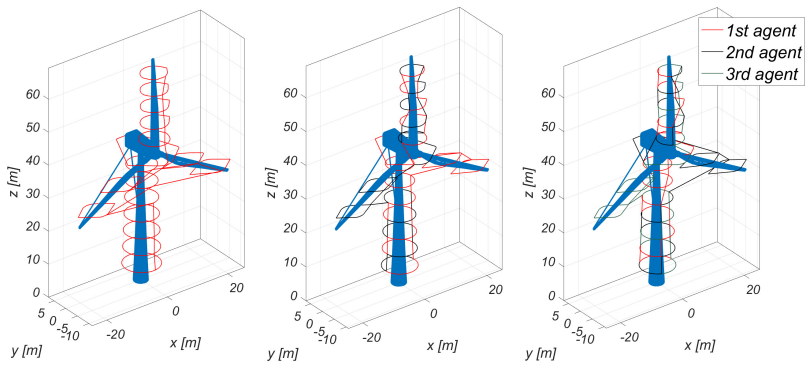
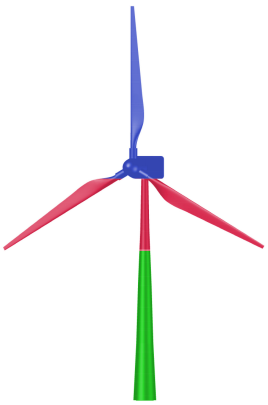


Figure 14: Generated path for different scenarios.

Figure 16 presents the yaw references, which have been provided for each agent in different scenarios. As the number of agents increases, the coverage task is completed faster thus the yaw changes more frequently, as is indicated by the provided simulation results.

To demonstrate the applicability of the method, two inspection scenarios have been performed. For the first scenario, an indoor artificial substructure was assembled as depicted in Figure 17. The structure consisted of 6 boxes with dimensions of $0.57 \times 0.4 \times 0.3\,\mathrm{m}$ with unique patterns and without any



g

Figure 15: Wind turbine with identified branches.

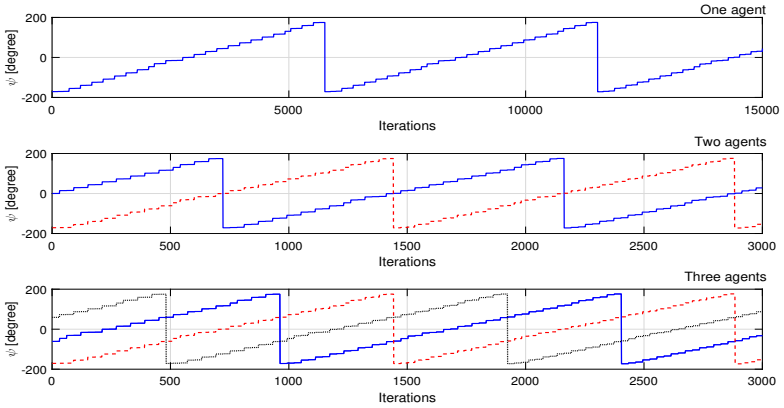


Figure 16: Yaw references for each scenario.

branches. In this case, two aerial agents were assigned to cover the structure. In these experiments, the Vicon Motion-capture (Mo-cap) system was used for the precise object localization. This information was utilized by the NEO for the autonomous flight. After the end of the experiment, the pose data from the Mo-cap system and the stereo stream were used by the mapping algorithm. The actual and the reconstructed structures are depicted in Figure 17.

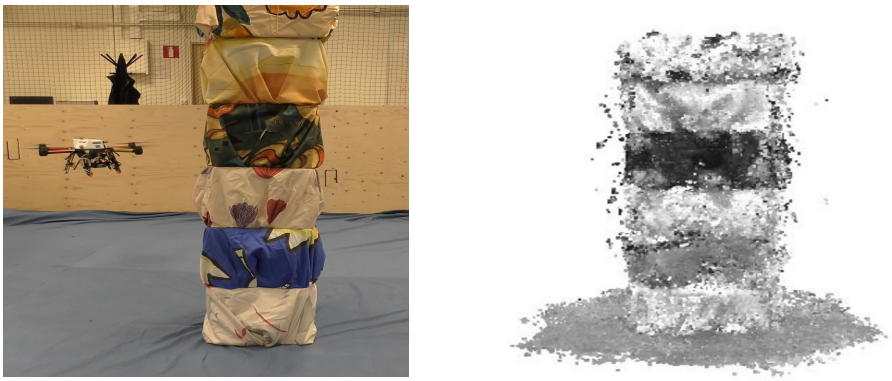


Figure 17: On the left is the simple indoor structure to be reconstructed and, on the right, the cooperative pointcloud of the structure.

Furthermore, Figure 18 presents the actual and reference trajectories that the two agents followed. Moreover, the Mean Absolute Error (MAE) is summarized in the following table 2.

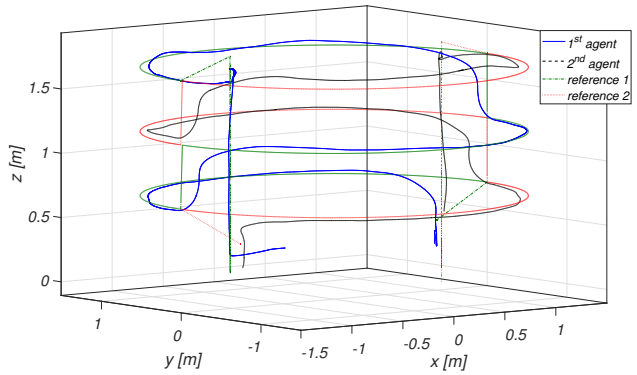


Figure 18: Trajectories which are followed in indoor experiment.

Table 2: Mean Absolute Error (MAE) of the trajectory followed by first and second agents for the indoor experiment.

MAE	x [m]	y [m]	z [m]
1^{st} agent	0.02	0.02	0.05
2^{nd} agent	0.03	0.02	0.06

Additionally, the starting position of each has 180 ° difference and the agents completed the mission in 166 s instead of 327 s. The average velocity along the path was 0.2 m/s and the points were fed to the agents in such a way as to guarantee the maximum distance and avoid collision.

To retrieve the 3D mesh of the structure, the Autodesk ReCap 360 was used [46]. ReCap 360 is an online photogrammetry software suited for accurate 3D modeling. The reconstructed surface obtained from the image data is shown in Figure 19.

Figure 19: Cooperative 3D mesh of the indoor structure.

To evaluate the performance of the method for a real autonomous inspection task, an outdoor experiment was conducted. For this purpose the Luleå University's campus fountain was selected to represent the actual infrastructure for the cooperative aerial inspection. The fountain has a radius of 2.8 m and height of 10.1 m without branches. Since in the outdoor experiments, motion

capturing systems are rarely available, the localization of the UAV relied only on the on-board sensory system, in order to achieve a fully autonomous flight. Thus, the UAVs followed the assigned paths with complete on-board execution, based on visual inertial odometry localization. The actual and reference trajectories followed by both platforms are depicted in Figure 20, while the MAE is provided in the Table 3.

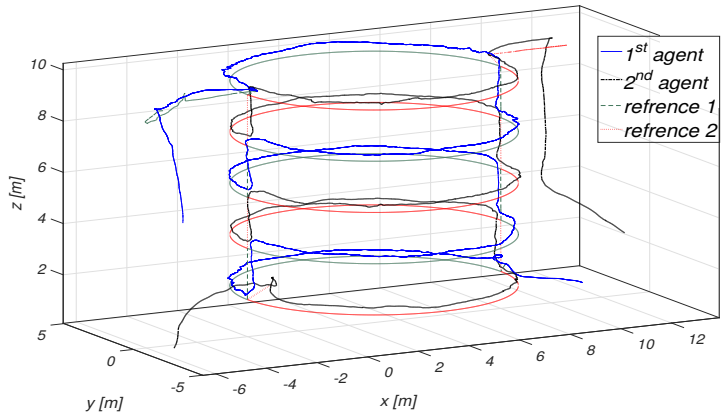


Figure 20: Trajectories which are followed in outdoor experiment.

Table 3: Mean Absolute Error (MAE) of the trajectory followed by first and second agents for the outdoor experiment.

MAE	x [m]	y [m]	z [m]
1^{st} agent	0.19	0.21	0.29
2^{nd} agent	0.21	0.23	0.31

For the reconstruction, the image streams from both aerial agents were combined and processed by the SfM algorithm as described in Section 3. The fountain and its sparse 3D model are presented in Figure 21.

In the proposed experiment the same strategy as the indoor experiment is followed for two agents. The starting position of each of them has the maximum

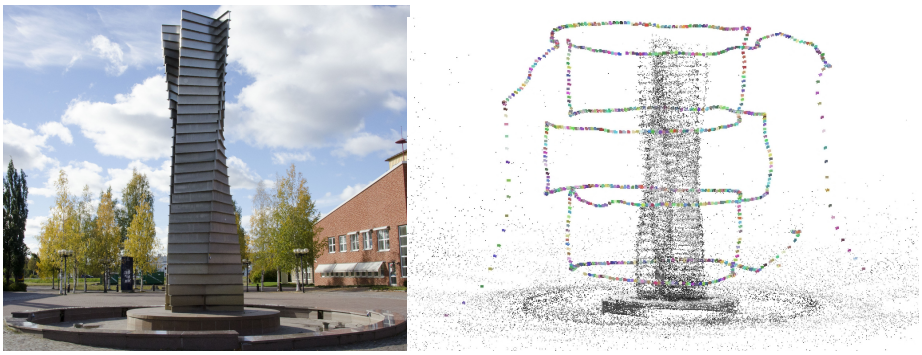


Figure 21: On the left is the Luleå University's outdoor fountain, and, on the right, the cooperative pointcloud of the structure with estimated flight trajectories.

of distance with 180 ° difference. The overall flight time is reduced from 370 s to 189 s and the average velocity along the path was 0.5 m/s. The sparse reconstruction provided in Figure 21 cannot be used for inspection tasks, since it lacks texture information and contains noise. Similarly to the indoor experiment, the reconstructed surface obtained from image data is shown in Figure 22. The results show that the collaborative scheme of the path planner could be successfully integrated for automating inspection tasks (https://youtu.be/IUfCrDHL-Dc).

4.3. Quantification of the Design Choices

In this article, three different inspection scenarios were studied: Case 1) simulation results from a wind turbine inspection, Case 2) an inspection of artificial indoor structure, and Case 3) a real-world outdoor infrastructure inspection. In all the cases there have been multiple limitations and challenges. The initial a priori information for enabling the aerial inspection of infrastructure needed is to have a 3D model of the structure. In cases that a non-detailed information can be obtained (e.g. blueprint or CAD model), as has been presented, the infrastructure can be approached by connected simple geometrical shapes without a loss of generality.

For all these experimental test cases and from a practical point of view there

Figure 22: Cooperative 3D mesh of the outdoor structure.

are multiple limitations mainly due to: 1) the experimental setup and 2) the surrounding environment. The limitations in the experimental setup are mainly related to the utilized hardware and software that can directly affect the inspection mission. In the proposed experiment the following components are critical for the successful mission: 1) localization, 2) control configuration, 3) visual processing and 4) flight time. Additionally, from a practical perspective, important factors that influence the aerial coverage task in field trials are the heavy wind gusts (e.g. near wind turbines) that distort the followed path, direct sunlight that immediately degrades the performance of the vision algorithms, and increased humidity levels and low temperatures that cause malfunctions in the platform electronics. Finally the need for a complete no rain environment and temperatures above +4C should be highlighted. Except for these limitation factors, later, the available design choices and the corresponding effect on the results will be discussed.

Safety distance: the safety distance between the agents and the infrastructure is a critical factor for the experiment, as this selection directly affects the safety of the mission and the overall quality of the obtained images. For safety reasons

and collision avoidance, it is preferable to have a proper safety margin from the structure. This margin can provide enough time for the safety pilot to avoid collisions in case of an accident. Additionally, the larger value of the Ω results in a larger camera footprint area. This has a direct effect on creating shorter paths for the UAVs that are now capable of covering bigger areas. However, the larger the value for Ω is, the more the quality of the images is reduced. This can be overcome by utilizing better cameras with a higher resolution. In the case of the indoor experiment, the safety distance was set to 1 m, as the flying area was limited, while in the outdoor experiment the Ω was set as 3 m, as there are extra disturbances to the system (such as wind gust) and the localization system was accurate in a few centimeters.

Field of view: this parameter is dependent on the hardware that will be used in the experiment and in most cases it is considered and modeled as a constant. This value directly affects the camera footprint. To be more specific, the larger the FOV value is, the larger the camera footprint area becomes. However, the FOV and the camera resolution has a direct impact on the image quality. In this article a camera with a visual sensor with a 78° FOV was used.

Number of agents: the number of agents has a direct relation to the size of the structure and the flight time of the platform. For example, in the case of the wind turbine inspection it is almost impossible to do the inspection with one agent as the flight time is 25 min. Thus, multiple agents are assigned for the inspection and the flight time is reduced to 12 min for each agent in the case of three agents. From another point of view, the more the agents there, the longer time the optimization algorithm takes for creating the multiple UAV paths.

Controller: In this article MPC is used as the position controller. There are multiple parameters that can affect the performance of the control scheme such as: 1) weights of the objective function terms, since the larger value of the weights are, the smaller the error in those terms are. To be more specific, in the case of the position error in the objective function, a higher weight results in a smaller positioning error, and 2) constraints: each platform has physical constraints such as maximum velocity, maximum acceleration, bound on pro-

pellers velocities, boundaries on angles, etc. During the experiments, it was observed that in the case of the outdoor trials, mainly due to the existence of the wind gusts, the MPC should have larger bounds on angles and thus these were changed from 15 ° to 45 °, which resulted in a better path tracking and wind gust compensation. It should also be highlighted that the error in the position and orientation may result in a different camera footprint area from the desired one, with a direct impact on the quality of the achieved inspection and reconstruction.

Velocity of the agent: this parameter is selected based on the experiment and hardware setup. The velocity of the agent should not be larger than the camera frames rate, which will result in blurred images. Additionally, the velocity of the UAV should be higher than the wind gust in order to compensate for it and due to safety reasons the velocity should be also reasonable for the safety pilot to be able to take control of the UAV in unpleasant scenarios. Thus, in this article the velocity is chosen as 0.2 m/s for thr indoors experiment and 0.5 m/s for the outdoors experiment.

Overlapping: Multi-view algorithms require the processing of the same surface, captured in multiple frames from different viewpoints, to provide a proper reconstruction. More specifically, a substantial overlapping between frames is considered. The absolute minimum overlapping required for each observed 3D point is only 2 frames. For increased robustness, practical implementations consider at least 3 images. In the presented manuscript the generated paths in the experimental trials considered at least 3 overlapping images for the reconstructed points. Additionally, 40% overlapping between slices was chosen.

Number of features: The basis of the SfM algorithms is mainly the correspondence search among frames. The correspondence step includes feature extraction (distinctive points in 2D frame), feature matching and additional verification to filter out outliers. Generally, the feature extraction step identifies a high number of features, e.g. 500, but they are decreased after the matching and validation steps. Therefore, the number of inlier features in every frame is not fixed and varies depending on the extraction algorithm, the occlusions, and the object

geometry. In the performed experimental trials, a mean value for every frame was around 200-300 features.

Camera frames: Multi-view algorithms can handle increased frame rates with proper view selection, when considering the data capturing. For the indoors and outdoors experimental trials the frame rate was fixed at 20 Hz. However, this parameter is highly connected with the image resolution, where with a higher resolution the algorithms can reconstruct in more details, since there exist fewer ambiguous matches in the captured frames. Generally, a higher resolution is preferred over higher frame rates.

Clustering: as it has been discussed in Section 2.2.2, in addition to the *K-means*, the MIP clustering was also investigated in the presented test cases. Thus, all the calculations, as stated before, have been performed on a computer with an Intel Core i7-6600U CPU, 2.6GHz and 8GB RAM and were implemented in MATLAB. In each scenario, both methods were evaluated and the computation time is provided in Table 4. In the first case the number of points in each slice is between 500 and 8000, and the number of clusters is 1 or 3. The average computation time for *K-means* was 0.2 s and the average computation time for the MIP clustering method with CPLEX solver and heuristic approaches were 75 s and 80 s respectively. The computation time of the MIP clustering method can be reduced further by increasing the processing power, down sampling the structure, or utilizing different optimization solvers. In the second and third cases there was only one branch and the number of points in each slice was between 60 and 150. Overall, the MIP clustering method provides a global solution but the computation power is still the main restriction to real-life applications. However, if the utilization of MIP is not causing problems in the realization of the proposed approach, this method could be utilized for performing the necessary clustering.

Table 4: Computation time for clustering methods in different scenarios.

Cases	*K-means*		MIP CPLEX solver		MIP heuristic approach	
	avg.	max	avg.	max	avg.	max
1	0.2 s	0.45 s	75 s	100 s	80 s	170 s
2	0.10 s	0.24 s	2.4 s	3.4 s	5.3 s	16.2 s
3	0.14 s	0.32 s	3.5 s	4.2 s	6.6 s	18.1 s

5. Conclusions

This article addresses the C-CPP for the inspection of complex infrastructures by utilizing multiple agents. The novelty of the proposed scheme stems from the establishment of an overall framework for the path planning of multiple UAVs to solve the coverage problem in complicated structures. A mathematical framework for solving the coverage problem by the introduction of branches and safety distances in the algorithm has been established. The established theoretical framework provides a path for accomplishing a full coverage of the infrastructure, without simplification of the infrastructure (number of considered representation points), in contrast to many existing approaches that simplify the infrastructure to an area of interest and solve it by various optimization methods; methods that could not be applied otherwise due to the inherent NP-hard complexity of the problem. In addition, this article has demonstrated the direct applicability and feasibility of coverage by multiple UAVs in field trials, while indicating the pros and cons when performing real life tests. This effort, integrated and adapted fundamental principles from control, image processing, and computer science, in a fully functional and efficient approach. Furthermore, the extended field trials, the comparison with the indoor experiments and the sharing of knowledge during the field trials in the form of lessons learned is another outcome of this work. From an application point of view, this article has contributed in presenting a complete cooperative aerial coverage system that can be directly applied in any kind of infrastructure, with the reported limitations.

[1] European Investment Bank, Public and private financing of infrastructure, Evolution and economics of private infrastructure finance, 2010.

[2] R. A. Fernandes, Line-mounted, movable, power line monitoring system, uS Patent 4,904,996 (Feb. 27 1990).

[3] N. Metni, T. Hamel, A uav for bridge inspection: Visual servoing control law with orientation limits, Automation in construction 17 (1) (2007) 3–10.

[4] M. Burri, J. Nikolic, C. Hürzeler, G. Caprari, R. Siegwart, Aerial service robots for visual inspection of thermal power plant boiler systems, in: Applied Robotics for the Power Industry (CARPI), 2012 2nd International Conference on, IEEE, 2012, pp. 70–75.

[5] P. Cheng, J. Keller, V. Kumar, Time-optimal uav trajectory planning for 3d urban structure coverage, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2008, pp. 2750–2757.

[6] K. Alexis, G. Nikolakopoulos, A. Tzes, L. Dritsas, Coordination of helicopter UAVs for aerial Forest-Fire surveillance, in: Applications of Intelligent Control to Engineering Systems, Springer Netherlands, 2009, pp. 169–193.

[7] E. Galceran, R. Campos, P. Edifici IV, N. Palomeras, P. de Peguera, D. Ribas, M. Carreras, P. Ridao, Coverage path planning with realtime replanning and surface reconstruction for inspection of 3d underwater structures using autonomous underwater vehicles.

[8] B. Englot, F. S. Hover, Sampling-based coverage path planning for inspection of complex structures.

[9] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robotics and Autonomous Systems 61 (12) (2013) 1258–1276.

[10] S. S. Mansouri, P. Karvelis, G. Georgoulas, G. Nikolakopoulos, Remaining useful battery life prediction for UAVs based on machine learning, in:

20th World Congress of the International Federation of Automatic Control (IFAC), 2017.

[11] L. Zuo, W. Yan, R. Cui, J. Gao, A coverage algorithm for multiple autonomous surface vehicles in flowing environments, International Journal of Control, Automation and Systems 14 (2) (2016) 540–548. doi:10.1007/s12555-014-0454-0.

[12] P. N. Atkar, H. Choset, A. A. Rizzi, E. U. Acar, Exact cellular decomposition of closed orientable surfaces embedded in R^3, in: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, Vol. 1, IEEE, 2001, pp. 699–704.

[13] E. Galceran, R. Campos, N. Palomeras, M. Carreras, P. Ridao, Coverage path planning with realtime replanning for inspection of 3d underwater structures, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 6586–6591.

[14] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, R. Siegwart, Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots, Autonomous Robots (2015) 1–20.

[15] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, J. Valente, Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots, Journal of Field Robotics 28 (5) (2011) 667–689.

[16] A. Adaldo, S. S. Mansouri, C. Kanellakis, D. Dimarogonas, K. Johansson, G. Nikolakopoulos, Cooperative coverage for surveillance of 3d structures, IROS.

[17] Galceran, Carreras, Planning coverage paths on bathymetric maps for in-detail inspection of the ocean floor, IEEE, 2013, pp. 4159–4164.

[18] D. B. West, et al., Introduction to graph theory, Vol. 2, Prentice hall Upper Saddle River, 2001.

[19] S. E. Schaeffer, Graph clustering, Computer science review 1 (1) (2007) 27–64.

[20] J. Nocedal, S. J. Wright, Sequential quadratic programming, Springer, 2006.

[21] L. A. Wolsey, Mixed integer programming, Wiley Encyclopedia of Computer Science and Engineering.

[22] B. Sağlam, F. S. Salman, S. Sayın, M. Türkay, A mixed-integer programming approach to the clustering problem with an application in customer segmentation, European Journal of Operational Research 173 (3) (2006) 866–879.

[23] P. Hansen, B. Jaumard, Cluster analysis and mathematical programming, Mathematical programming 79 (1-3) (1997) 191–215.

[24] W. L. G. Koontz, P. M. Narendra, K. Fukunaga, A branch and bound clustering algorithm, IEEE Transactions on Computers 100 (9) (1975) 908–915.

[25] B. S. Everitt, S. Landau, M. Leese, D. Stahl, Hierarchical clustering, Cluster Analysis, 5th Edition (2011) 71–110.

[26] S. Lloyd, Least squares quantization in PCM, IEEE transactions on information theory 28 (2) (1982) 129–137.

[27] I. I. CPLEX, V12. 1: Users manual for cplex, International Business Machines Corporation 46 (53) (2009) 157.

[28] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Transactions on Mathematical Software (TOMS) 22 (4) (1996) 469–483.

[29] X.-Z. Liu, J.-H. Yong, G.-Q. Zheng, J.-G. Sun, An offset algorithm for polyline curves, Computers in Industry 58 (3) (2007) 240–254.

[30] K. Alexis, G. Nikolakopoulos, A. Tzes, Model predictive quadrotor control: attitude, altitude and position experimental studies, IET Control Theory & Applications 6 (12) (2012) 1812–1827.

[31] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, W. Burgard, An evaluation of the rgb-d slam system, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 2012, pp. 1691–1696.

[32] P. J. Besl, N. D. McKay, Method for registration of 3-d shapes, in: Robotics-DL tentative, International Society for Optics and Photonics, 1992, pp. 586–606.

[33] M. Labbé, F. Michaud, Memory management for real-time appearance-based loop closure detection, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 1271–1276.

[34] J. Sivic, A. Zisserman, Efficient visual search of videos cast as text retrieval, IEEE transactions on pattern analysis and machine intelligence 31 (4) (2009) 591–606.

[35] J. L. Schönberger, J.-M. Frahm, Structure-from-motion revisited, CVPR, 2016.

[36] C. Wu, Towards linear-time incremental structure from motion, in: 2013 International Conference on 3D Vision-3DV 2013, IEEE, 2013, pp. 127–134.

[37] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[38] D. Nistér, An efficient solution to the five-point relative pose problem, IEEE transactions on pattern analysis and machine intelligence 26 (6) (2004) 756–770.

[39] X.-S. Gao, X.-R. Hou, J. Tang, H.-F. Cheng, Complete solution classification for the perspective-three-point problem, IEEE transactions on pattern analysis and machine intelligence 25 (8) (2003) 930–943.

[40] Robot Operating System (ROS).
URL http://www.ros.org/

[41] E. Fresk, S. S. Mansouri, C. Kanellakis, G. Nikolakopoulos, Reduced complexity calibration of mems imus, in: 25th Mediterranean Conference on Control and Automation (MED), 2017.

[42] S. Lynen, M. Achtelik, S. Weiss, M. Chli, R. Siegwart, A robust and modular multi-sensor fusion approach applied to mav navigation, in: Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), 2013.

[43] K. Alexis, G. Nikolakopoulos, A. Tzes, Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances, Control Engineering Practice 19 (10) (2011) 1195–1207.

[44] M. Kamel, T. Stastny, K. Alexis, R. Siegwart, Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system, in: A. Koubaa (Ed.), Robot Operating System (ROS) The Complete Reference, Springer, (to appear).

[45] M. Bloesch, S. Omari, M. Hutter, R. Siegwart, Robust visual inertial odometry using a direct ekf-based approach, in: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, 2015, pp. 298–304.

[46] Autodesk RECAP 360.
URL http://recap360.autodesk.com/