Vision-based Self-Assembly for Modular Quadrotor Structures

Yehonathan Litman, David Saldaña, Vijay Kumar

Abstract—Modular aerial swarms offer promising advantages due to their ability to perform tasks unsuitable for individual quadrotors. Plenty of work has been done in developing proper control algorithms for modular swarms, but little has been done in regards to assembly and control for quadrotors in outdoor environments with only onboard sensors. In this paper, we present both a docking and assembly methodologies aimed at constructing modular aerial formations with quadrotors sporting only a single camera and an Inertial Measurement Unit (IMU). With the usage of a realistic simulation environment, we verify that our control algorithm for docking performs well under extraneous forces. In addition, we derive an algorithm that addresses the modules’ limited sensing capabilities in order to assemble a wide variety of structures. We also prove that our assembly algorithm’s time complexity is low, showing that it has potential in being used on large scale modular robotic systems.

I. INTRODUCTION

Large aerial robots are difficult to maintain, require plenty of storage space, more expensive to build, and less agile. On the other hand, flying robotic modular swarms may offer the same abilities but without the cons associated with large individual aerial robots. Multiple small yet agile individual robots can quickly get to a location of interest, whether indoors or outdoors, and assemble in order to do tasks such as rapid infrastructure repair and building [1]–[3], high weight cargo lifting [4], [5], and search and rescue [6]–[8].

Many works on modular aerial swarms present different ways of giving modular swarms the ability to self-assemble, but certain challenges still remain. One of these is the detection when two modules are docked together. Docking is usually detected either using external motion capture systems such as Vicon [1], [9], or via physical means such as magnetic connectors that are either equipped with sensors or complete a circuit connection upon docking [6], [10]. Some recent related works did away with these sensors and external infrastructures and aimed to make a control formulation that uses only vision for docking as well as other tasks such as perching [11]–[14]. These methods are based on visual servoing [15], which was also used in our previous work [16] for facilitating docking in modular aerial swarms. However, while visual servoing enables docking in many kinds of robots, they rest on the assumption that the robots can be controlled and are not affected by external forces. This issue can be particularly problematic in aerial robots, who are considerably affected by external forces from the environment in which they operate.

The other problem is that when forming structures with visual servoing, the camera has a limited field of view and cannot guide the robots to form structures from any arbitrary angle. Some works [13], [18], [19] address this limitation and integrate proper structural forming using only vision, but are limited to ground robots and/or underwater docking. Thus, there has been scarce work done on integrating visual servoing with forming structures using modular quadrotor robots.

Our first contribution in this work is presenting a docking strategy that combines the use of a camera and an Inertial Measurement Unit (IMU) while taking into account the magnetic force between the modules in order to calculate the ideal force to apply while docking. This way, the modules can avoid environmental forces that may throw them off track when they are in the process of docking, leading to less failed docking procedures and a quicker overall assembly. In addition, it avoids having to use any external infrastructure to detect whether the robots have docked together.

We also present a recursive, efficient, and flexible algorithm that can calculate a good placement of each aerial module all while accounting for the robotic modules’ limited...
field of view. Our algorithm also accounts for physical and theoretical weaknesses in order to not only assemble a viable structure, but also to account for physical weaknesses in the theoretical structure and correct them by picking a robot furthest away from the center of mass. This robot will guide the entire structure in future dockings, so by being placed in the border of the structure it guarantees that the structure’s field of view will be clear and could be used for future dockings. Lastly, we show in our run-time analysis that our algorithm is suited for assembling very large robotic aerial structures.

II. ModQuad-VI Model

We first present the terminology and main concepts behind our modular aerial system.

Definition 1. (Module) A module, $M$, is a flying robot that can move by itself in a three dimensional environment and horizontally dock to other modules.

All modules are based on a quadrotor platform equipped with a cuboid cage of width $w$ and height $h$, with the dimensions $w \times w \times h$. Each module has the same mass $m$, shape, inertia, and actuators. There are four magnets on each face of a module for a total of 16 magnets. We can define a formation of rigidly attached modules as a structure.

Definition 2. (Structure) A structure, $S = \{M_1, \ldots, M_N\}$, is a non-empty set of rigidly connected modular aerial robots that behave as a single rigid body. These modules are horizontally connected by docking along the sides so the resulting shape has the same height $h$.

Our quadrotors are based on our previous design [16], we equip each module with three vertical WhyCon tags [17] on the left, right, and back sides of the cage. Each module is also equipped with a fisheye camera providing images at VGA resolution. Modules perform docking based on the camera input, so a module uses its camera to follow the tag on another module. An example of a docking procedure can be seen in Fig. 2 and it can be inferred that the docking procedure is a directed operation (i.e. the camera of one module must be pointing at the tag of another module). As such, one module must hover at a fixed position so the other can track its tag and then initiate the docking procedure. We can define these modules as the Waiting module and the Docking module, respectively.

Definition 3. (Waiting module) A Waiting module, $M_w$, is a module that hovers at a fixed position in space.

Definition 4. (Docking module) A Docking module, $M_d$, is a module that is moving towards the Waiting module horizontally in order to dock.

Lastly, we describe some of the dynamics and coordinate frames in our work. The inertial coordinate frame, $W$, is fixed and has its $z$-axis pointing upwards. The location of the center of mass of the $i$th module $M_i$ in relation to the inertial frame is $x_i, y_i, z_i \in \mathbb{R}^3$. The linear velocity and acceleration are represented by $\dot{x}_i$ and $\ddot{x}_i$. In the structure’s coordinate frame $S$, the origin is located at the center of mass of the structure. In addition, the position of $M_i \in S$ is denoted as $[x_i, y_i, z_i]^T$. In our work, we assume that we can directly control the velocities and accelerations. For details about our dynamics model, we refer the reader to [1].

Fig. 2 illustrates two modules that are about to dock, where $M_1$ is docking to $M_2$ as the module that will guide the structure in future dockings, which would be $M_2$ in our case. The position of the structure in $W$ can be denoted by $\hat{x}_S \in \mathbb{R}^3$.

III. Visual-Magnetic Docking Control

We employed magnetic connectors in order to dock two aerial structures, but only for physically joining modules together. We abstained from using any additional sensing hardware in favor of detecting the docking, and we only use measurements from the IMU. In addition, we take into account the perturbations from external forces such as wind. If these forces remain unaccounted for, they can misalign the docking module’s trajectory towards the waiting module’s tag before docking is completed, causing the entire process to fail. Therefore, we present a control algorithm that compensates for outside influences during docking by including the magnetic attraction between two modules.

First, we calculate the force $F$ on a magnet in the docking module in relation to the position difference $\hat{p}$ with a magnet on the waiting module

$$F(\hat{p}) = \frac{3\mu_0 |\hat{m}|^2}{4\pi|\hat{p}|^r}[2\hat{m}(\hat{m} \cdot \hat{p}) + \hat{p}(||\hat{m}||^2 - 5||\hat{m} \cdot \hat{p}||^2)],$$  

where $\hat{p}$ is the unit vector of $p$, $\hat{m}$ is the magnetic moment with $\hat{m}$ as its unit vector, and $\mu_0$ is the vacuum permeability constant [20]. We also calculate the torque $\tau$ due to
the position difference between two magnets [21]

\[
\tau = \frac{3\mu_0 |\mathbf{m}|^2}{4\pi |\mathbf{p}|^3} (\mathbf{m} \cdot \dot{\mathbf{p}})(\mathbf{m} \times \dot{\mathbf{p}}), \tag{2}
\]

and these calculation of force and torque is done for each of the four magnets on the docking module’s side. We could include the effect of each magnet’s moment on the other as a cross product between the two, but we assume this quantity to be close to zero given that the orientation of the magnets during docking is nearly the same. Given that (1) and (2) are nonlinear, we can linearize the force and torque using a Jacobian matrix \( \mathbf{J}_{\mathbf{F}, \tau} \) that differentiates the forces with respect to the position difference and magnetic moment [22]

\[
\begin{bmatrix}
\mathbf{F} \\
\dot{\mathbf{p}} \\
\dot{\mathbf{m}}
\end{bmatrix} = \mathbf{J}_{\mathbf{F}, \tau}
\begin{bmatrix}
\mathbf{p}
\\
\mathbf{m}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial \mathbf{F}}{\partial \mathbf{p}} & \frac{\partial \mathbf{F}}{\partial \mathbf{m}} \\
\frac{\partial \dot{\mathbf{p}}}{\partial \mathbf{p}} & \frac{\partial \dot{\mathbf{p}}}{\partial \mathbf{m}} \\
\frac{\partial \dot{\mathbf{m}}}{\partial \mathbf{p}} & \frac{\partial \dot{\mathbf{m}}}{\partial \mathbf{m}}
\end{bmatrix}
\begin{bmatrix}
\mathbf{p}
\\
\mathbf{m}
\end{bmatrix}, \tag{3}
\]

For our purposes, we are only interested in the partial derivatives of the force and torque in relation to the position difference. Those can then be used to compute the desired partials \( \dot{\mathbf{F}}^\star_{\mathbf{p}}, \dot{\mathbf{r}}^\star_{\mathbf{p}} \) of force and torque in respect to position during docking for all four magnets on the docking module’s side

\[
\dot{\mathbf{F}}^\star_{\mathbf{p}} = \mathbf{K}_{\mathbf{rF}} (\dot{\mathbf{F}} - \dot{\mathbf{F}}_{\mathbf{p}, \mathbf{T}}), \tag{4}
\]

where \( \dot{\mathbf{F}}_{\mathbf{p}, \mathbf{T}} \) is the desired partial of the magnetic force in relation to position, and \( \dot{\mathbf{F}}_{\mathbf{p}} \) is the control input that represents the actual partial. Since the docking module should be as still as possible, the desired torque is zero

\[
\dot{\mathbf{r}}^\star_{\mathbf{p}} = \mathbf{K}_{\mathbf{rT}} (-\mathbf{r}_{\mathbf{p}}), \tag{5}
\]

where \( \mathbf{r}_{\mathbf{p}} \) is the control input and actual partial of magnetic torque in relation to position, and \( \mathbf{K}_{\mathbf{rF}}, \mathbf{K}_{\mathbf{rT}} \) are the proportional gains of our controller. Given the small time frame when docking takes place, we can use the accelerometer measurements to detect jumps or peaks in the measurements. When a jump is detected, we assume that a docking between two modules is successfully completed. To avoid erroneous detections, we only look for jumps in acceleration when docking takes place, we can use the accelerometer

\[
|| \int_{t_i}^{t_j} (\mathbf{a}_M - \mathbf{g}) \, dt || > c_F,
\]

where \( t_i, t_j \) denote the most recent time interval in the IMU timeframe, and \( c_F \) is a threshold that the accelerometer jump integration needs be above in order to be detected as a docking. \( c_F \) is defined in terms of the magnetic force in an ideal docking. As such, \( c_F \) is the output of (1) given a very small position difference and dividing the result by the mass of a single module. We consider torque to be negligible since we assume a perfect docking to be perfectly linear.

IV. Vision-based Self-Assembly

As defined in section [1], when two modules dock to assemble a structure, one condition has to hold true: the docking module’s camera has to be tracking (or following) one of the waiting module’s tags. In abstract terms, during the docking procedure, the docking module’s front-side (i.e. where the camera is situated) is directed at one of the sides of the waiting module that is not the front. This considerably limits the options for docking procedures. We must consider the orientation of the modules before docking in a manner that prevents the resulting or sub-structures from forming a “deadlock,” i.e. when the structure either has no available cameras on any module that can be used for further docking, or is physically incompatible with other structures. As shown in Fig. [2], this can theoretically happen in two ways: either the formed structure is in a loop configuration and each camera points at a tag, leaving no cameras that can perform future dockings, or the formed structure has a misaligned module, barring the possibility to form the desired structure.

Fortunately, we can use the concept of a self-avoiding walk (SAW) [23], [24] to generate a path from one vertex to another which never self-intersects. A SAW is a path on a graph (which in our case, is on a lattice in \( \mathbb{R}^2 \) with nearest-neighbour edges) that does not visit the same vertex twice. In our case, we consider a vertex in a graph to be a module in a structure. Using our previous work [25] we generate positions for each module such that they do not intersect with each other, yet we are not interested in the positions generated by the path, but the apparent direction of the path. We do this by giving five vertices to each module, one in the center to signify the module’s position and four on all of its sides. For each docking, the docking module’s position and camera vertices, as well as the waiting module’s tag and position vertices all become connected through a single path. A SAW must be generated such that the camera vertex of one module in the structure, the master’s, remains unconnected, but for all other modules there must be a path from its position vertex, through its camera, and to the master’s position vertex. For all modules in a structure excluding the master, \( M \in S \setminus \{M^\star\} \), there must exist a desired path \( P_M \) from the module’s position vertex to the master’s position vertex, where \( P_M = \{M_p, M_c, \ldots, M_{p}^\star\} \), and \( M_c, M_t, M_p \) are the camera, tag, and position vertices, respectively.

This strategy theoretically works for all structure configurations, yet there are physical implications which need to be accounted for. These include cases where structures loop, physically impeding the possibility of assembly, or structures that are unstable to fly.

After generating the position for each module in the structure, we must find the master which will guide the whole structure for future dockings. If we naively generate a SAW, it may work for simple cases where no module is connected to more than two other modules. In that case, structures would be organized and formed in a snake-like fashion; i.e. tail-to-head, where the master would be the head.
Fig. 3: A success and failure in forming the desired loop structure in (a); (b) Proper assembly of the desired structure, with a visual example of a SAW from one of the modules in the structures to the master represented by the black line. (c) A misalignment in camera position occurred in the smaller structure, so assembly cannot be completed, (d) No further cameras are available, rendering the structure unusable. In the figure, the grey squares are quadrotor modules, the black vertices represent the tag vertices, white vertices the position vertices, and the blue squares are cameras, also representing the camera vertices. The rotors of all modules are oriented in relation to the camera, so if the camera is rotated by 90 degrees, then so are the rotors.

(5)

\( M^* = \text{argmin}_{M_i \in S} \text{DEG}(M_i) \) (7)

where \( \text{DEG} \) is a function that returns the degree of each module. It is plausible for the optimal solution to be a set of modules of the same degree, so we can simply choose the median element of the set, and if the set is even the median can be randomly chosen between the two middle values. The module that is least connected is likely at the boundary of the structure, guaranteeing that its field of view is free of obstructing modules and available for future dockings.

Algorithm 1 SAW Generation

**Input:** A master or virtual master \( M^* \)

**Output:** A dictionary set of module orientations \( X \)

**function** GENERATESAW \((M^*)\)

\[ X \leftarrow \emptyset \]

\[ \text{for } M_i \in \text{NEIGHBORS}(M^*) \text{ do} \]

\[ M_i \leftarrow \text{NEXTNEIGHBOR}(M_i) \]

\[ \text{if } \text{UNORIENTED}(M_i) \text{ is true then} \]

\[ X \leftarrow X \cup \text{CALCULATEORIENTATION}(M_i) \]

\[ X \leftarrow X \cup \text{GENERATESAW}(M_i) \]

end if

end for

return \( X \)

end function

After choosing a master we generate SAWs from the master’s origin to the modules whose degree is the least in a head-to-tail, depth first search fashion. We calculate the orientations of each module such that \( P_M \) exists, and generate a tree with branches connected in parallel recursively as in Alg. 1. In addition, once a module’s orientation with is calculated using \( \text{CALCULATEORIENTATION} \) its orientation cannot be modified again. This will be important when generating looping structures. Our algorithm generates the path head-to-tail, and then continues to build the structure head-to-tail. Thus, our head is guaranteed to be placed in a location with no obstructions.

### B. Looping Scenario

The algorithm for the snake scenario works just as well when a structure has loops. However, when we choose the master via (7), we must choose a module that is at the corner of a loop. We need to account for the symmetry of the structure, or else generating the tails for a loop structure with no modules in the middle would be physically flawed and the structure could not be fully assembled. For any such structure, there would always be two tails generated in parallel because all modules have two neighbors that would be used as virtual masters to generate tails from. A looping structure with no modules in the middle always contains an even number of modules \( N \). If we pick off a master and then 2 generate tails in parallel, we get \((N - 1) \mod 2 = 1\), or one module that will be docking to the structure last. Since our structure is made of square modules, it will be symmetric, thus the position of the last module to dock is the mirror of the master across the center of the structure. Due to the fact that a module with a camera can only dock to a structure if its back side is unobstructed, picking a master that
Fig. 4: Configurations for assembly, where the missing spot cannot be filled in (a), but can be filled in (b). The docking in (b) may be performed by moving diagonally into position, but the docking in (a) can simply not be performed. The robots are represented by the shaded squares and the missing spot is represented by a dashed line. The cameras on the modules are represented by the small solid blue rectangles. The two tails generated in parallel from the master are colored by the time steps they docked (i.e. blue modules docked at the first time-step, green modules at the second time-step, etc.).

is not the corner would place the last module at a position where its back side is obstructed, but picking a master that is at the corner would leave its back side unobstructed. This is similar to cases in [1], [7], since multiple modules attempt to dock at the same time. We demonstrate desirable and undesirable pre-assembly configurations in Fig. 4 with the desired structure from Fig. 3.

V. EXPERIMENTAL VALIDATION

We used the open-source Robotic Operating System (ROS) [26] for developing the tethered capsule platforms software. For conducting the experiments, we simulated the quadrotors using MAVROS in Gazebo. Given Gazebo’s open-source physics engine, we opted to realistically simulate magnetic force in relation to the position difference of two modules so that our simulation could accurately replicate experiments in the real world. We equipped each quadrotor with virtual tags and a camera, in addition to a protective cage and simulated quadrotor hardware. We also placed 4 magnets per side and gave each the same magnetic moment through parameters in the Gazebo environment.

A. Visual-Magnetic Control

To verify that our visual-magnetic aided control algorithm worked properly, we aligned the orientations of both the waiting and docking module with that of the inertial frame so that the docking module would dock to the back of the waiting module. We used this simple docking configuration in order test out how well the docking module could calculate the magnetic forces during the docking procedure. With MAVROS simulating environmental forces such as wind in Gazebo, we tested for the force and torque calculated by the docking module. The linear and angular accelerations due to linear force and torque, respectively, are shown in Fig. 5. Note that the figure shows the forces calculated for one of the magnets on the docking module, but the forces for each of the four magnets were calculated and applied to our dynamics.

B. Assembly

Here we demonstrate as well as scrutinize the advantage of our assembly algorithm. First, we experimented with five modules in order to demonstrate the assembly of a structure that demonstrated our assembly’s algorithm ability to work in realistic scenarios. We show an example of the assembly process in Fig. 1 and the reader may note that the white points are the cameras of the modules, but they also conveniently represent the camera vertices described in section IV. In addition, the reader may notice that \( P_M \) exists for all modules.

However, in the future we intend to use our algorithm to make structures that will consist of many more modules than just five, therefore we will analyze the time complexity of our proposed assembly approach. We first present four different configurations in order to demonstrate the algorithm’s ability to generate different and complex structures. The configurations are shown in Figures 6a - 6d, and consist of up to 40 robot modules. The configurations consist of a square landing platform, an hourglass formation, a complex structure with a hole in the middle, and a bridge network. As expected, all structures can assemble without a problem. In Figures 6e - 6h the docking sequences are shown and we the time it took for a complete assembly is visualized through modules having the same color at certain time-steps. The reader may notice that our algorithm “ripples” throughout the structure, especially in Fig. 6c as it gives an impression that it assembles a layer by layer of the structure. This is because our algorithm recursively calculates the orientation.
for neighbours, so the more connected a structure is the faster it will be assembled. Fig. 6 represents the best case scenario where the structure is most connected and symmetric, and our algorithm requires four time steps in total, or one more than the square root of the modules in the structure. As the number of modules $n$ in the structure approach infinity, we see that our algorithm would run in $O(\sqrt{n})$ time. The time-steps taken for a most connected structure is simply the number of nodes in the path from the master to the node mirrored to it, minus the master node. However, in the worst case scenario we have a structure which is simply a line. In that case the master would be at the edge, and assembling would take $O(n)$ time since the algorithm would be bottlenecked and its recursiveness would not help. In addition, parallelization cannot take place. Each module’s orientation would be calculated at a time step, because all modules would have one neighbour, excluding the one they are being oriented towards.

VI. CONCLUSION

In this work, we showed that it is possible to make a modular aerial swarm that can self-assemble into complex structures without external input. We addressed the quadrotor modules’ camera direction limitation by devising an efficient assembly algorithm, in addition to deriving a docking algorithm that addresses unwanted forces during docking. Because we only used a simulation environment for this paper, we plan to construct quadrotors in order to conduct experiments outdoors and test our work further. In addition, we hope to better analyze the time complexity of our assembly algorithm, as well as search for faster assembly approaches that parallelize the assembly procedure even more. However, given that the assembly time for the structure is also dominated by the time it takes to dock, the assembly time for structures in real life may be significantly different than in simulation. As such, we hope to explore the possibilities of faster docking. Lastly, we also hope to test for assembly using quadrotor modules equipped with more than one camera.

REFERENCES


