sensors

Article

Development of an Autonomous Unmanned Aerial Manipulator Based on a Real-Time Oriented-Object Detection Method

Shijie Lin [1,†], Jinwang Wang [1,†], Rui Peng [1] and Wen Yang [1,2,*]

1 School of Electronic Information, Wuhan University, Wuhan 430072, China; linshijie@whu.edu.cn (S.L.); jwwangchn@whu.edu.cn (J.W.); pengrui@whu.edu.cn (R.P.)
2 State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan 430079, China
* Correspondence: yangwen@whu.edu.cn
† These authors contributed equally to this work.

Received: 5 March 2019; Accepted: 23 May 2019; Published: 25 May 2019

Abstract: Autonomous Unmanned Aerial Manipulators (UAMs) have shown promising potential in mobile 3-dimensional grasping applications, but they still suffer from some difficulties impeding their board applications, such as target detection and indoor positioning. For the autonomous grasping mission, the UAMs need ability to recognize the objects and grasp them. Considering the efficiency and precision, we present a novel oriented-object detection method called Rotation-SqueezeDet. This method can run on embedded-platforms in near real-time. Besides, this method can give the oriented bounding box of an object in images to enable a rotation-aware grasping. Based on this method, a UAM platform was designed and built. We have given the formulation, positioning, control, and planning of the whole UAM system. All the mechanical designs are fully provided as open-source hardware for reuse by the community. Finally, the effectiveness of the proposed scheme was validated in multiple experimental trials, highlighting its applicability of autonomous aerial rotational grasping in Global Positioning System (GPS) denied environments. We believe this system can be deployed to many potential workplaces which need UAM to accomplish difficult manipulation tasks.

Keywords: aerial manipulation; aerial system; deep learning

1. Introduction

During the past several decades, Unmanned Aerial Vehicles (UAVs) have shown surprising potential among numerous applications [1,2]. When mounted with different kinds of sensors, the UAVs can extend the sensing range and transform a static sensing task into a mobile sensing task. The UAVs' advantages of freely using 3-dimensional (3D) space bring possibilities to many traditional tasks, like manipulation tasks in a warehouse or a factory.

Unmanned aerial manipulators (UAMs) are known as one specific type of UAVs equipped with one or multiple robotic arms and have attracted a lot of research interests in recent years [3]. One main advantage of a UAM is that it shows promising potential in transforming passive sensing missions into mobile 3D interactive missions, like grasping [4] and assembling [5]. Aerial maneuvering and hovering capabilities make it possible for a UAM to accomplish many kinds of missions that are difficult for human workers, such as grasping plastic bottle waste on a cliff. Furthermore, there are many places, like an Amazon warehouse, with the potential to deploy a UAM system for autonomous picking and placing. To fully utilize the space in a warehouse, goods can be placed at a relatively high place, which is difficult for human workers to reach. At this time, the UAM can give a lot of help. In future warehouses, there will be many UAMs flying simultaneously to sort the goods. Besides, places, like the Fukushima nuclear power plant in Japan, that are filled with obstacles and are hard for human workers to work inside are ideal places to deploy a UAM. Potential applications of UAMs can easily come from people's imaginations and surely not be limited to the situations mentioned above, which is the reason why we believe UAMs have a bright future.

For autonomous aerial manipulation missions, building a controllable system is always the first step. However, many factors can influence the stability of the overall system, such as the change of Center of Gravity (CoG) generated by the movements of the robotic arm, the reaction force produced by the robotic arm, and the complex aerodynamics effects. Many efforts have been made to reduce these effects [6–10]. A comprehensive dynamic model of hexacopter with a robotic arm has been built in [6], analyzing the effects of CoG change and mass distributions. The kinematics and dynamics models of a quadrotor with n Degree of Freedom (DoF) robotic arm have been formulated in [7]. A Variable Parameter Integral Backstepping (VPIB) controller proposed in [8] can control a UAM with better performance than a Proportional Integral Derivative (PID) controller. With a movable compensation mechanism, a multilayer architecture controller that compensates for the internal and external effects, layer by layer, to control the UAM was presented [9]. To suppress the torque generated by the movements of the robotic arm, a novel mechanism with a simplified model has been adopted in [10]. For some scenarios where multiple UAMs are required to work at the same time, some swarm studies [11,12] are also worth considering.

Usually, human operators are unable to accurately control the UAM due to the point of view change and data transmission delay. Also, for the monocular camera, the lack of depth perception in teleoperation makes it hard for the operators to decide whether the object is appropriately put into the claw of robotic arm. Such inaccuracy causes the robotic arm to be difficult to align and grasp the objects. Therefore, it is better the UAM can work without relying on external control by human operators. However, if there is no external control by human operators, the UAM needs to recognize the objects by itself, meaning we need to give the UAM perception ability. Currently only a few works [13–15] considered robust visual perception aided autonomous grasping. An Image-Based Visual Servo (IBVS) was implemented in [13] to help to locate the position of the targeted object. Feature models were used in [14] to find the position of known targets. In [13], correlation filters were adopted to track targets.

In recent years, end-to-end object detection algorithms like [16–18] have shown surprising results. Faster R-CNN [16] is an end-to-end two-stage framework for object detection. It firstly uses the Region Proposal Network (RPN) to extract proposals which contain objects, then classifies and refines these proposals. You Only Look Once (YOLO) [17] is a one-stage object detector, and it is much faster than two-stage framework. Considering efficiency, SqueezeDet [18] fully utilized SqueezeNet [19], which is a small network but achieves AlexNet [20] level accuracy with 50x fewer parameters. And SqueezeDet achieves a good tradeoff between detection accuracy and efficiency.

However, attempts to directly apply these methods in the UAM system usually end with failure or unsatisfactory robustness. The reason lies in the fact that objects, like bottles, in the UAV perspective often appeared with arbitrary poses, as shown in Figure 1a, since the camera is tight-coupled and rotated with the UAV body. Moreover, most grasping missions need a target's rotation angle for not touching the target when the end-effector is approaching. Furthermore, performance, like robustness and efficiency of the detection algorithms, plays an important role in grasping missions since it can bring more mobility to the UAM. However, current common oriented bounding boxes-based methods presented in [21,22] are designed for large-scale objects detection, like detecting a car in a large satellite image. Since the cars only make up a few pixels in a large satellite image, they need a much deeper convolution network to improve their accuracy. And this deep network cannot run in real-time performance even with an NVIDIA GTX1080 graphic card. Considering the size and weight requirement of a drone platform, the NVIDIA Jetson TX2 is the current optimal choice for mobile deep learning platforms. But methods like [21,22] are still unable to run onboard since both require more than 11G memory to load a deep convolution network, and the Jetson TX2 only has an 8G memory, let alone the real-time performance. Without a real-time performance, the grasping efficiency has been greatly reduced because the UAM needs to fly much slower to not pass the target.

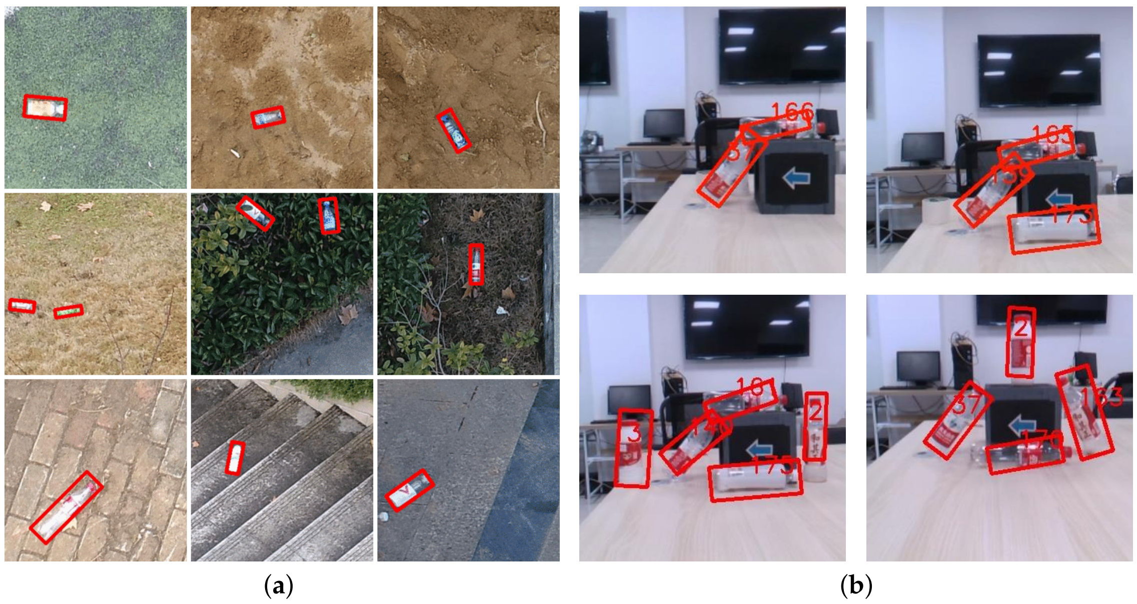


Figure 1. (**a**) Sample images in Unmanned Aerial Vehicle (UAV)-Bottle Detection (BD) dataset with plastic bottle groundtruth labeled by red box. (**b**) Detection results of Rotation-SqueezeDet with 2D box and rotation angle of multiple objects at the same time. Oriented angles of bottles are shown in the images.

In order to solve the problems mentioned above, we proposed Rotation-SqueezeDet, which can regress rotation angle and position in a 2-dimensional (2D) image in near real-time. Unlike the common horizontal bounding box descriptor (c_x, c_y, w, h), where (c_x, c_y) is the center location, and w and h are the width and height of the bounding box, respectively, Rotation-SqueezeDet introduces a new θ term, and thus uses (c_x, c_y, w, h, θ) to describe object position and rotation angle in the 2D image. This not only makes the detection more robust since the bounding box included fewer background, but also provides the rotation angle of the target. By using an Intel RealSense D435 depth camera, the relative 3D distance of targets can be measured in the point cloud generated by registered depth image once the target is detected. Hence, a rotation-aware grasping for autonomous grasping is possible. A glimpse of detection results is shown in Figure 1b.

Now we want to deploy this detection in the UAM for autonomous grasping. However, the system mentioned in previous paragraphs have only considered neither the control problem or system compensate problem but an overall applicable indoor system. Usually, they test the system in an outdoor environment with strong Global Positioning System (GPS) signals for localization or use an expensive external visual caption system like VICON (www.vicon.com) to provide state estimation of the drone. Both of these two ways are hard to achieve in a real industrial environment. So, we adopted the lightweight Simultaneous Localization and Mapping (SLAM) system VINS-Mono [23] for indoor state estimation, giving our system the ability to fly indoors with only an inexpensive camera. Furthermore, based on the Rotation-SqueezeDet, we designed and implemented a complete indoor, workable UAM system.

The main contributions of this paper can be summarized in two folds. First, the Rotation-SqueezeDet method was proposed. This method can run on Jetson TX2 in near real-time and enable successful rotation-aware grasping. Moreover, we believe that this method will be suitable in not only the aerial grasping but also in more general missions. Second, we have designed and assembled the whole UAM platform and fully tested this system in an indoor environment. The system is affordable since it costs less than $2300 USD and can fly without relying on expensive visual motion caption system.

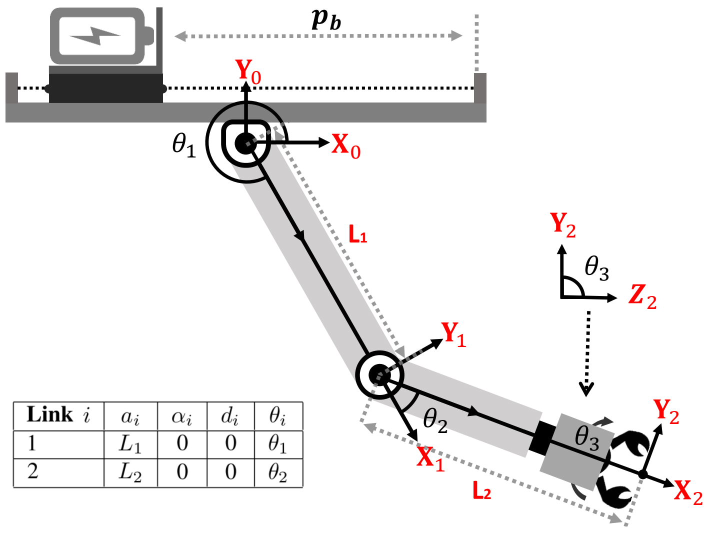
We believe this system can be adopted to many potential workplaces, like Amazon warehouses, which need UAMs to accomplish difficult manipulation tasks. All mechanical structures are provided as open-hardware for reuse by the community (https://github.com/eleboss/UAMmech).

The rest of this paper is organized as follows. Section 2 describes the design and details of the overall system. Section 3 describes the formulation, control, and planning of the UAM system. Section 4 describes the complete vision system including Rotation-SqueezeDet. Section 5 experimentally demonstrates the system including autonomous grasping and vision system performance in an open dataset. Finally, the conclusion and future work are presented in Section 6.

2. System Description

2.1. Notation

The East, North, and Up (ENU) coordinate system are used as a world-fixed inertial frame corresponding to $\{x_w, y_w, z_w\}$. Following the definition of well known Denavit–Hartenberg (D–H) parameters [24], the link frame of Link i is defined as the $\{x_i, y_i, z_i\}$. Especially, we define $\{x_0, y_0, z_0\}$ as the fixed arm frame. The definition of the link frame is detailed in Figure 2 and the table in the bottom left gives the D-H parameters of the robotic arm. The body frame is assumed to be the geometrical center of the UAM denoted as $\{x_b, y_b, z_b\}$. G_x, G_y, G_z indicate the CoG of the UAM in body frame $\{x_b, y_b, z_b\}$. (ϕ, θ, ψ) indicate the roll-pitch-yaw Euler angels. $\{x_t, y_t, z_t\}$ define the detected targets in the RealSense D435 camera frame.



Link i	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2

Figure 2. The coordinate system and Denavit–Hartenberg (D–H) parameters of the robotic arm.

2.2. Hardware

In this work, a modified DJI hexacopter frame F550 (www.dji.com/flame-wheel-arf/feature) was adopted. The hardware components are shown in Figure 3.

The propulsion module of the UAM was composed of Sunnysky (www.rcsunnysky.com) x3108s motor, Hobbywing (www.hobbywing.com) platinum Electronic Speed Controller (ESC), and a 10-inch propeller. The Pixhawk4 (www.holybro.com/product/55) autopilot with PX4 (https://github.com/PX4/Firmware) V1.8.0 flight stack and NVIDIA Jetson TX2 (developer.nvidia.com/embedded/buy/jetson-tx2) were placed at the top of the UAM as the main computing devices. A global shutter monochrome camera (www.jinyandianzi.com) was tight-coupled with the Pixhawk using a 3D-printed anti-vibration damping plate. A Benewake TFmini (www.benewake.com/tfmini.html) laser rangefinder was chosen for being cheap, lightweight, and with up to 12 m maximum detection range, and was mounted downward facing to provide altitude feedback. An Intel RealSense D435 (www.realsense.intel.com/stereo) camera was mounted at the middle of the drone, facing forward to find the targets.

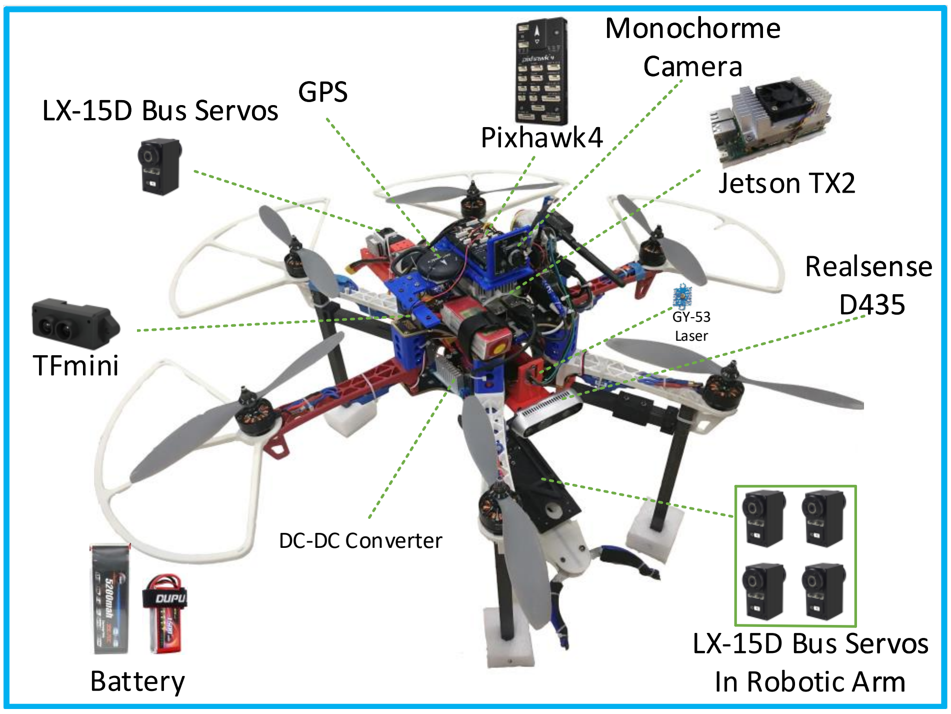


Figure 3. Hardware components of the Unmanned Aerial Manipulator (UAM).

We built a Displacement Compensation System (DCS) that can move counterweight to align the CoG and thus improves the stability of the total system. The DCS was mounted in the middle of UAM and made by 3D printed Poly Lactic Acid (PLA) material, including tow rails, a slide table, and a bus servo to provide drive force. The LEBOT (www.lobot-robot.com) LX-15D serial bus servos were chosen for being budget-friendly, lightweight, and having multiple extra structures to facilitate the installation. Another significant advantage of the bus servo is it can largely reduce the wiring complexity and control difficulty. So we could only use one serial port in the Jetson TX2 to control all servos. While the LX-15D servo can only provide 240° feedback, we used a short-range Time of Flight (ToF) laser rangefinder GY-53 to provide the battery position feedback.

A 5200 mAh 4S-35C battery weighted 0.525 kg was used for providing enough power to the propulsion system and as a counterweight for the DCS. And this battery could sustain the flight time around 10 min. Another 1500 mAh 3S-30C weight 0.138 kg battery was used for providing power to the robotic arm and computing facilitates.

Drive forces of the robotic arm were also provided by LX-15D servos. The robotic arm had 3-DoF and could grasp objects by the end-effector. The first 2-DoF provided the robotic arm mobility to move at the planar 2D plane. The last DoF enabled a rotational grasping by cooperating with the vision system. The last DoF was of vital importance for the oriented object grasping, since the unrotated grasping could easily tip the object.

The total takeoff weight of the UAM was about 4.08 kg. Thanks to the carbon fiber material and PLA material, the robotic arm was only weighted 0.459 kg and had 43 cm extended range.

2.3. Software Architecture

The software runs on two main processors: Pixhawk4 and Jetson TX2, and all processes are running onboard. Figure 4 gives an overview of the system software architecture. Note that the Jetson TX2 is overclocked to run at the maximum clock rate and unlocked two external cores to generate more computing power.

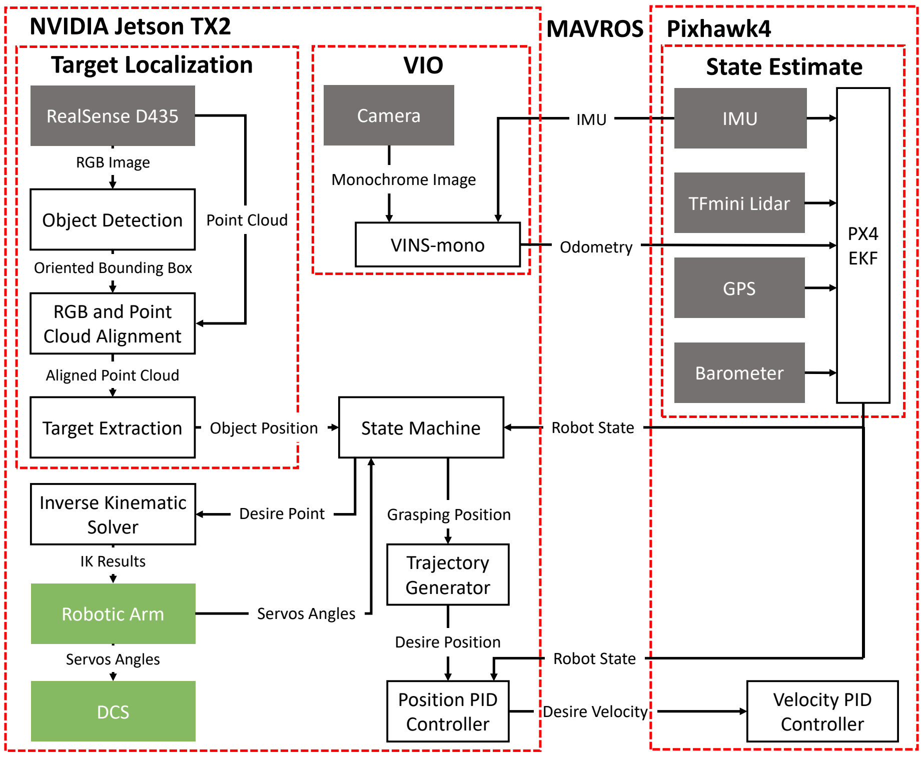


Figure 4. An overview of the software architecture. PID = Proportional Integral Derivative; IMU = Inertial Measurement Unit.

The Robot Operating System (ROS) [25] is a pseudo-operating system which allows developers to work cooperatively by following its running mechanism. Modules related to state estimation and flying control are running on the Pixhawk4 and exchange data with the Jetson TX2 by MAVROS (https://github.com/mavlink/mavros). Visual Inertial Odometry (VIO) and object detection algorithms are running on the Jetson TX2. A state machine is adapted to set the robotic arm motion and UAM waypoint. So, the grasping position (x^s_w, y^s_w, z^s_w) of the UAM can be given by:

$$\begin{bmatrix} x^s_w \\ y^s_w \\ z^s_w \\ 1 \end{bmatrix} = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} + T^W_B T^B_C \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} - T^W_B T^B_0 \begin{bmatrix} x^g_0 \\ y^g_0 \\ 0 \\ 1 \end{bmatrix}, \tag{1}$$

where $T^W_B, T^B_0, T^B_C \in \mathbb{R}^{4\times4}$ are the homogeneous transformation matrix, T^B_0 transforms the fixed arm frame to body frame, T^B_C transforms the camera frame to the body frame, T^W_B transforms the body frame to the world-fixed frame, and (x^g_0, y^g_0) is the grasping point.

The standard cascaded position-velocity control of the hexacopter is well implemented inside the Pixhawk4 fly controller. More detailed information can be found at this link (https://dev.px4.io/en/flight_stack/controller_diagrams). We adopted this control strategy and well tuned its inside parameters to get a robust performance.

2.4. State Estimation

State estimation is the foundation of our system, as it provides crucial information to help other parts to achieve the best performance. In this work, we use the Pixhawk4 built-in Extended Kalman Filter (EKF) to fuse multiple sensors' feedback for state estimation.

GPS usually fails to provide global positioning feedback when in an indoor environment. Hence, in order to fly indoors without using expensive visual motion caption system, we integrated the

VINS-mono VIO to provide the local position feedback, and it can provide the highest level of accuracy and robustness compared with multiple VIO [26]. The VINS-mono runs at 10hz with loop-closure and the output is rotated to ENU world-fixed frame denoted (x_w^v, y_w^v, z_w^v). For the VINS-mono, we used the Inertial Measurement Unit (IMU) in the Pixhawk4 as the inertial input, and a monochrome global shutter camera ran at 640×400 resolution and 90 Frames Per Second (FPS) to provide clear images as visual input. To reduce drifting, the monochrome camera and Pixhawk4 are tight-coupled by the 3D printed structures, the camera intrinsic matrix and camera to IMU transformation parameters are carefully calibrated by using kalibr [27]. Due to the computation limitation and data transmission delay, the output of VINS-mono runs in Jetson TX2 has about 140ms delay compared with current IMU output. In order to synchronize these outputs, we first calculate the velocity of VINS-mono estimation $(\dot{x}_w^v, \dot{y}_w^v, \dot{z}_w^v)$, then apply some random movements to the UAM, so the delay time can be found by comparing $(\dot{x}_w^v, \dot{y}_w^v, \dot{z}_w^v)$ with the Pixhawk4 velocity estimation. Finally, (x_w^v, y_w^v, z_w^v) is used as external vision aid of the Pixhawk4 onboard EKF to give 100hz state estimation.

For robust flying, the main altitude feedback is not given by the VINS-mono but the TFmini rangefinder. The total delay of TFmini measurement is about 30 ms.

3. Robotic Arm & DCS

3.1. Robotic Arm Motion Planning

In order to find the workspace of robotic arm, the forward kinematics of a 3DoF robotic arm is given by the following equations:

$$\begin{aligned} x_0 &= L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ y_0 &= L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \\ \theta_3 &= \theta, \end{aligned} \tag{2}$$

where L_1, L_2 are the lengths of the first and the second links. θ_1, θ_2, θ_3 are the rotation angles of each joint, and (x_0, y_0) is the point in arm fixed frame. Since the angle θ_3 is only related to the target rotation angle θ, the workspace of our robotic arm is equal to a planar 2DoF robotic arm model. The table presented in Figure 2 gives a clear D-H parameters definition of the robotic arm.

When applying the UAM into real-world scenes, the flow generated by rotors can blow the lightweight targets away, like empty plastic bottles, and it is hard to predict whether an object can be easily blown away by the downward flow or not. Hence, a safety grasping action is better to be taken under weak flow influenced conditions. High-fidelity Computational Fluid Dynamics (CFD) simulation results of many different UAVs have been presented in [28]. From the observation of these results, the flow generated by each UAV's rotor is decreasing rapidly in the outer-wing area. Inspired by this observation, we planned the robotic arm to grasp in a weak flow area. Based on the flow measurements described in Section 5.2 and the actual mechanical movement range of robotic arm, the actual workspace is given in Figure 5, the green points indicate weak flow influenced area, and red points indicate strong flow influenced area. The blue star point is the dropping point and the purple star point is the grasping point (x_0^g, y_0^g). The robotic arm holds at the yellow star point (x_0^f, y_0^f) during flight. And all these points can be redefined depending on the applications.

We used Inverse Kinematics (IK) to solve the desired rotation angle of each joint. Assuming the robotic arm is planning to move to (x_0, y_0) point, the IK result is given by:

$$\theta_2 = \pm \arccos\left(\frac{{x_0}^2 + {y_0}^2 - {L_1}^2 - {L_2}^2}{2L_1L_2}\right), \tag{3}$$

Here, $\theta_2 \in [0, \pi]$ is the downward elbow solution, and θ_1 can be derived by:

$$\theta_1 = \arctan\left(\frac{{x_0}^2}{{y_0}^2}\right) - \arccos\left(\frac{{x_0}^2 + {y_0}^2 + {L_1}^2 - {L_2}^2}{2L_1\sqrt{{x_0}^2 + {y_0}^2}}\right). \tag{4}$$

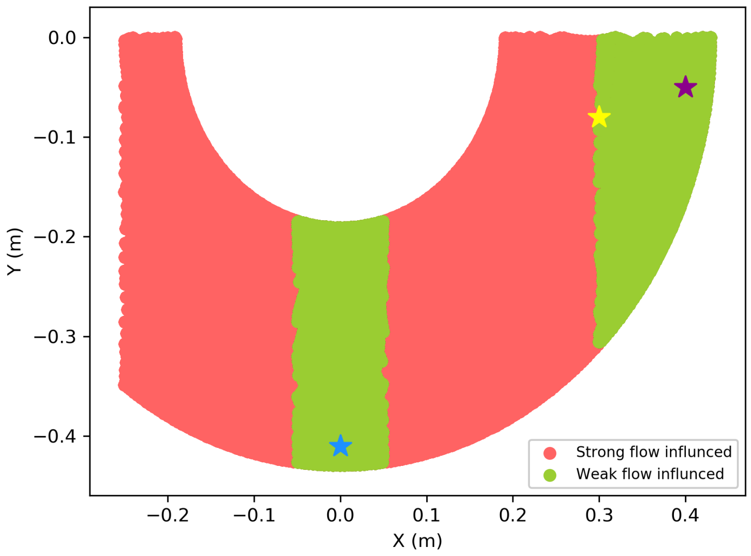


Figure 5. Workspace of robotic arm printed by using forward kinematics considered flow influence.

3.2. CoG Compensation

When the UAM is static on the ground and the robotic arm holds static at (x_0^f, y_0^f), a symmetry placement design is utilized to make sure the G_x and G_y are fitted with the geometry center.

However, movements of the robotic arm can change the G_x. For the dynamic G_x alignment, we adapted the strategy presented in [9] called DCS, moving the battery as a counterweight since its weight can provide sufficient compensation in the relatively short moving distance.

CoG transformation of Link i and the end-effector payload from link frame to the body frame are given by:

$$\begin{bmatrix} x_{bi}^g \\ y_{bi}^g \\ z_{bi}^g \\ 1 \end{bmatrix} = T_0^B T_i^0 \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \\ 1 \end{bmatrix}, \tag{5}$$

where $T_i^0, T_0^B \in \mathbb{R}^{4\times4}$ are the homogeneous transformation matrices, T_0^B transforms from fixed arm frame to body frame, and T_i^0 transforms from each link frame to the fixed arm frame. (x_i^c, y_i^c, z_i^c) is the CoG position of Link i in the fixed arm frame. $(x_{bi}^g, y_{bi}^g, z_{bi}^g)$ is the CoG position of Link i in body frame, and here $i = 3$ indicates the grasped object.

To align the G_x at geometry center, a linear slider is designed to move the battery and the position of the battery p_b in the body frame can be calculated by:

$$p_b = \frac{\sum_{i=1}^{3} m_i x_{bi}^g}{m_b}, \tag{6}$$

where m_i is the mass of Link i, m_b is the mass of battery.

The displacement compensation plays a key role in stabilizing the UAM. Without the DCS, the change of CoG can easily make aside rotor reach the maximum thrust, leading to it being unstable. And this method worked well in our system. To guarantee the compensation performance, we chose a reasonable range of rotation speed of each joint in a robotic arm to make sure it did not exceed the maximum compensation speed of the linear slider.

3.3. Control Strategy of Robotic Arm

The total control diagram of the robotic arm and DCS are shown in Figure 6. Here, $\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{p}_b$ is the rotation speed of the servos and the $\theta_{1cur}, \theta_{2cur}, \theta_{3cur}, p_{bcur}$ indicate the current feedback. Three PID controllers are adapted to control the robotic arm. Another PID controller uses the position of the battery p_b, detected by a laser rangefinder, to control the linear slider. All PID parameters are well tuned to guarantee stable and smooth control.

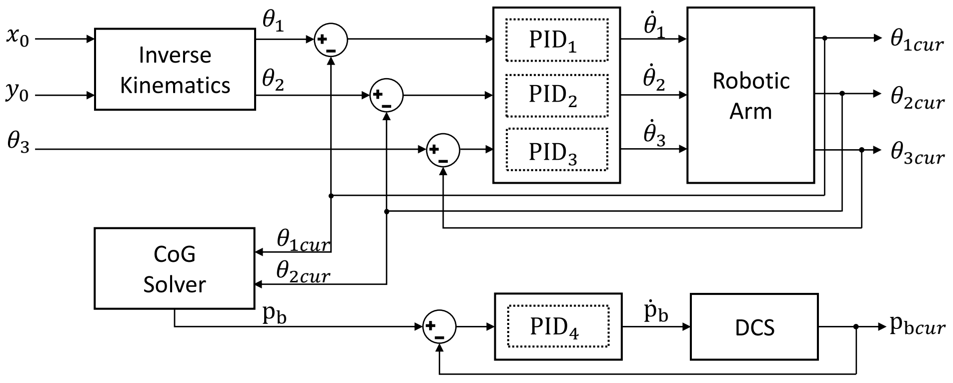


Figure 6. Control diagram of the Displacement Compensation System (DCS) and robotic arm. CoG = Center of Gravity.

4. Vision System

In this section, we introduce the vision system, including the light and fast oriented-object detection model called Rotation-SqueezeDet and the target localization framework based on Rotation-SqueezeDet detection results, and point clouds from the depth camera.

Our proposed model is inspired by SqueezeDet [18] and Rotation Region Proposal Networks (RRPN) [21]. As the former, SqueezeDet is a single-pass detection pipeline combining bounding box localization and classification by a single network. It appears to be the smallest object detector by virtue of a powerful but small backend network of SqueezeNet [19,29]. As the latter, RRPN is based on Faster R-CNN [16], but RRPN can detect oriented objects. The difference between Faster R-CNN and RRPN is described as below. In Faster R-CNN, the Region of Interests (RoIs) are generated by RPN, and the RoIs are rectangles which can be written as $R = (x_{min}, y_{min}, x_{max}, y_{max}) = (c_x, c_y, w, h)$. These RoIs have regressed from k anchors which are generated by some predefined scales and aspect ratios. Then these RoIs will be feed into RoI Pooling layer and some fully connected layers to obtain horizontal bounding boxes. However, in RRPN, instead, it uses Rotation anchors (R-anchors) and rotation RoI Pooling, which brings the ability to predict oriented bounding boxes denoted as $R = (c_x, c_y, w, h, \theta)$.

Object detection algorithms like Faster R-CNN have high accuracy but slow processing speed and large storage requirement. Moreover, RRPN is slower than Faster R-CNN, it takes twice as much time as the Faster-RCNN [21]. Thus, RRPN does not meet our run-time requirement. Considering a comparable accuracy and run-time on Jetson TX2, SqueezeDet is a suitable choice, it can run about 45FPS with 424 $\times$ 240 pixels image on Jetson TX2, and is easy to train. However, SqueezeDet cannot predict the θ of the rotated object because it uses horizontal bounding boxes. So, we designed a model which can generate oriented bounding boxes based on SqueezeDet and named Rotation-SqueezeDet, and it can run on Jetson TX2 in near real time.

4.1. Network Architecture

The overall model of Rotation-SqueezeDet is illustrated in Figure 7. In this model, a convolutional neural network, SqueezeNet V1.1, first takes an image as input and extracts a low-resolution, high dimensional feature map from the image. Then the feature map is fed into the $F_w \times F_h$ convolutional layers to compute oriented bounding boxes at each position of conv feature map. Next, each oriented bounding box is associated with $(5 + C + 1)$ values, where 5 is the number of bounding box parameters, C is the number of classes, and 1 is the confidence score. And each position on the convolution feature map computes $K \times (5 + 1 + C)$ values that encode the bounding box predictions. Here, K is the number of R-anchors, each R-anchor can be described by 5 parameters as $(c_x^a, c_y^a, w^a, h^a, \theta^a)$, (c_x^a, c_y^a) are R-anchor's center on image, w^a, h^a, θ^a are the width, height and angle of R-anchor, respectively.

$$\begin{aligned} v_x &= \frac{c_x - c_x^a}{w^a}, v_y = \frac{c_y - c_y^a}{h^a}, \\ v_w &= \log \frac{w}{w^a}, v_h = \log \frac{h}{h^a}, v_\theta = \theta - \theta^a + k\pi, \end{aligned} \tag{7}$$

where (c_x, c_y, w, h, θ) are parameters describe the predicted oriented bounding box, $(c_x^a, c_y^a, w^a, h^a, \theta^a)$ are parameters describe the R-anchor, and here $k \in Z$ to ensure $\theta \in [0, \pi)$.

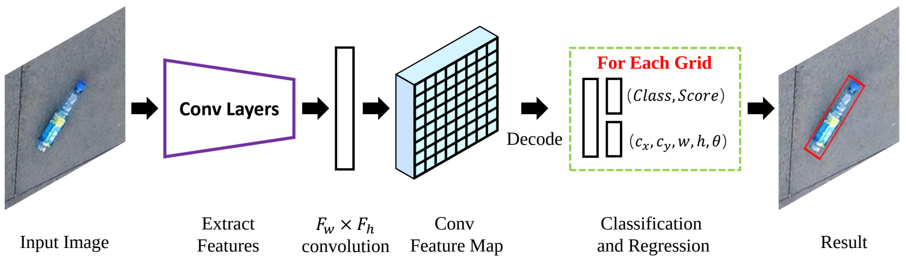


Figure 7. Rotation-SqueezeDet's detection pipeline.

4.2. Oriented IoU Computation

To efficiently predict the rotation angle of objects, we introduce a parallel Intersection over Union (IoU) computing method. First, we attempt to calculate the IoU using the OpenCV's functions *rotatedRectangleIntersection* and *contourArea* directly. However, the efficiency of these functions remains poor because they cannot compute parallel. Thus, we use a simple and efficient method to approximately compute the IoU in parallel, which is to use the angle deviation of two oriented bounding boxes. The approximate IoU [30] can be computed by:

$$\text{IoU}^\star = \text{IoU} * \text{abs}(1 - \frac{\theta_1^b - \theta_2^b}{\pi}), \tag{8}$$

where θ_1^b and θ_2^b are the rotation angle of two oriented bounding boxes, and IoU is computed by treating oriented bounding boxes as horizontal bounding boxes.

4.3. R-Anchors' Selection

The R-Anchors are different from the horizontal anchors. And for the R-Anchors' selection, we use a K-means based method described in [31] to select R-anchors' w and h to match the data distribution, we set k as 9 in K-means and treat objects' angle distribution as a uniform distribution, i.e., we set R-anchors' angle as $\{0, \frac{\pi}{9}, \frac{2\pi}{9}, \frac{3\pi}{9}, \frac{4\pi}{9}, \frac{5\pi}{9}, \frac{6\pi}{9}, \frac{7\pi}{9}, \frac{8\pi}{9}\}$. Therefore, there are 81 anchors at each conv feature map position.

4.4. Object Localization

To acquire the real world position of the target, we use the RGB-D camera to detect and locate the target. First, the aligned color image and point clouds can be obtained by aligning RGB image and depth image in the same coordinate system. Next, the subarea of the total point clouds containing location information of the target which can be extracted from the whole point clouds by utilizing the detection result (c_x, c_y, w, h, θ). After that, we use a small central subarea of target's point clouds to calculate its real-world position (x_t, y_t, z_t) in the camera frame given by:

$$(x_t, y_t, z_t) = \left(\frac{1}{L}\sum_{i=0}^{k^2}\mathbf{X}_p^i, \frac{1}{M}\sum_{i=0}^{k^2}\mathbf{Y}_p^i, \frac{1}{N}\sum_{i=0}^{k^2}\mathbf{Z}_p^i\right), \tag{9}$$

where $(\mathbf{X}_p, \mathbf{Y}_p, \mathbf{Z}_p)$ are the coordinates of the point clouds set of the center subarea of target's bounding box, its superscript indicates the i^{th} point value started from top left corner. L, M, N are the valid points number of $\mathbf{X}_p, \mathbf{Y}_p, \mathbf{Z}_p$, respectively. The central subarea of target's point clouds can be written as (c_x, c_y, k, k), the $k \times k$ is the size of the selected central subarea of target's point clouds, which can be calculated by:

$$k = \begin{cases} 5 & \text{if } \min(w, h) > 5 \\ \min(w, h) & \text{otherwise} \end{cases}. \tag{10}$$

Here, we set k as 5 to reduce the computing burden. Finally, the parameters of position and rotation angle of a target can be written as: (x_t, y_t, z_t, θ), θ is the rotation angle of target's anchor.

5. Experiments

The experimental setup is mentioned in Section 2. During the flight tests, all data are logged onboard with no external data transmission.

5.1. Vision System Results

We trained and evaluated our vision system based on our pervious work UAV-Bottle Detection (BD) [32], a bottle image dataset under UAV perspective. It contains about 34,791 object instances in 25,407 images labeled by oriented bounding boxes. For training and evaluating our model, 64% of the images were randomly selected as the training data, 16% as validation data, and the rest 20% as the testing data.

All object detection experiments were implemented on TensorFlow [33]. We used the pertained model, SqueezeNet v1.1, to initialize the network. And the system was trained 100k steps with a batch size of 20 and a learning rate of 0.01. Besides, weight decay and momentum were 0.0001 and 0.9, respectively. The optimizer was *MomentumOptimizer*.

As shown in Figure 8, the Average Precision (AP) of Rotation-SqueezeDet on UAV-BD is about 78.0% when IoU $= 0.5$. In Figure 9, we visualize the detection results in UAV-BD dataset. Our algorithms can be given two results; first is the common horizontal bounding box, marked by red color, without any rotation, and second is the oriented blue bounding box inside the red bounding box. We can clearly see that the blue boxes are more accurate than the red boxes since it contains fewer background pixels and locates the object more precisely.

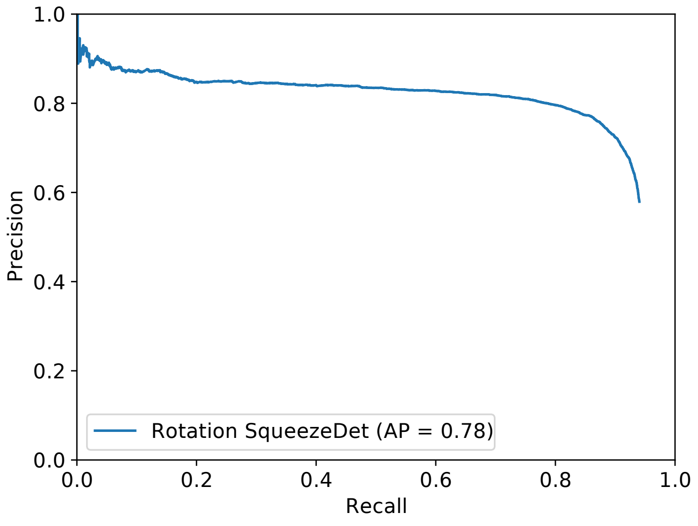


Figure 8. Average Precision (AP) results of Rotation-SqueezeNet validated in UAV-BD.

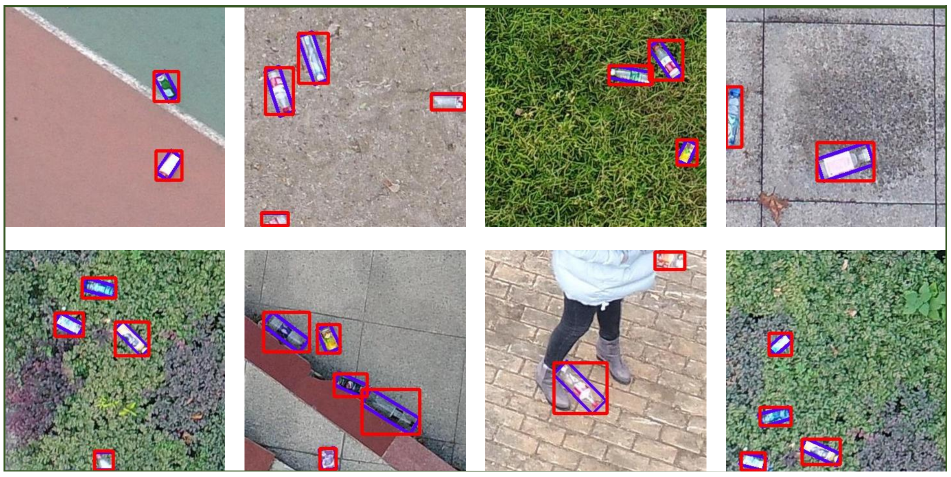


Figure 9. Detection results of the proposed detection algorithm in UAV-BD dataset. We demonstrate results in subfigures with varied backgrounds like grassland, playground and ground surface. Noticed that our algorithm could give the common bounding box in red and oriented bounding box in blue at the same time.

To analyze the performance of time consumption, we deployed Rotation-SqueezeDet on three different platforms and recorded the average processing time with 424 × 240 pixels color image. The detailed information of the deployed platform is shown in Table 1, and the results are shown in Table 2. From Table 2, we can see that our algorithm has achieved near real-time performance in the Jetson TX2 and real-time performance on both the Laptop and Server platform. Notice that our algorithm could run up to 111 FPS with a Titan Xp GPU, and it can save a lot of time during the training of Rotation-SqueezeDet.

Table 1. Details of platforms using in the processing time test.

	Laptop	**Server**	**Jetson TX2**
Central Processing Unit (CPU)	i7-6700HQ @ 2.6 GHz	E5-2620 V4 @ 2.10 GHz	Cortex-A57 @ 2 GHz
Graphics Processing Unit (GPU)	GTX 960M	GTX Titan Xp	256-core Pascal

Table 2. Processing time of Rotation-SqueezeDet in different platform.

	Laptop	Server	Jetson TX2
Model Processing Time (s)	0.017	0.005	0.0035
NMS Processing Time (s)	0.003	0.004	0.006
Overall Processing Time (s)	0.020	0.009	0.041
Overall FPS	50	111	24

The time consumption of Rotation-SqueezeDet is mainly coming from two parts. The first part is the processing of detection model based on SqueezeNet. The second part is the processing of Non-maximum Suppression (NMS). The processing speed of detection model is related to GPU, and the processing speed of NMS is related to CPU. So, using a more powerful GPU can speed up the detection model processing, and a more powerful CPU will process the MNS much faster.

5.2. Flow Distribution Validation

In order to give a rough parameter estimation for the planning mentioned in Section 3, we disarmed the UAM and keep it flying at a certain height about 1.5 m above the ground. Then we used a Smart AS856 (en.smartsensor.cn/products_detail/productId=248.html) anemometer to measure the downward flow distribution. Since the flow is highly complex, we cannot give a very accurate flow distribution by just using an anemometer; thus, we only measured a 1D flow distribution to give a rough estimation for the robotic arm planning. The moving path of the anemometer is given in the top right corner of Figure 10 and the vertical distance from the anemometer to the rotor is about 16cm to represent the overall rough distribution below the UAM.

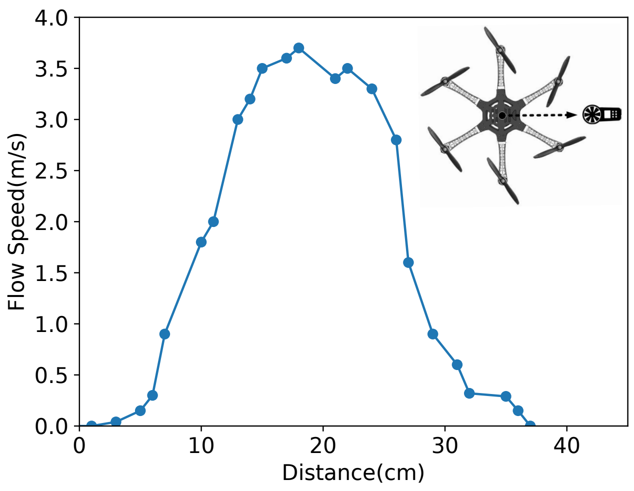


Figure 10. The flow speed measured by digital anemometer and the subfigure in the top right corner is the path of the anemometer during measurement.

Following this path, we recorded the average value of 5 measurements at one point and drew the curve in Figure 10. The horizontal axis is the distance from the central position of the UAM body to the outside following this path. From the observation of Figure 10, the flow speed is decreasing rapidly at about 8 cm and 30 cm, and reaching the top speed at about 21 cm which is the underside of rotors. The 8 cm is close to the inside of the UAM, and the 30 cm is close to the outside of the UAM. So, the flow speed is relatively weak when at the central position and outside position of the UAM.

5.3. Aerial Robotic Arm Moving Test

To evaluate the stability and controllability of our system, the UAV was programmed to hovering at a certain point. In Figure 11, the control errors of the system during this flight were logged and presented. After finishing the takeoff procedure, the robotic arm will automatically move to multiple points and rotate the end-effector 90°.

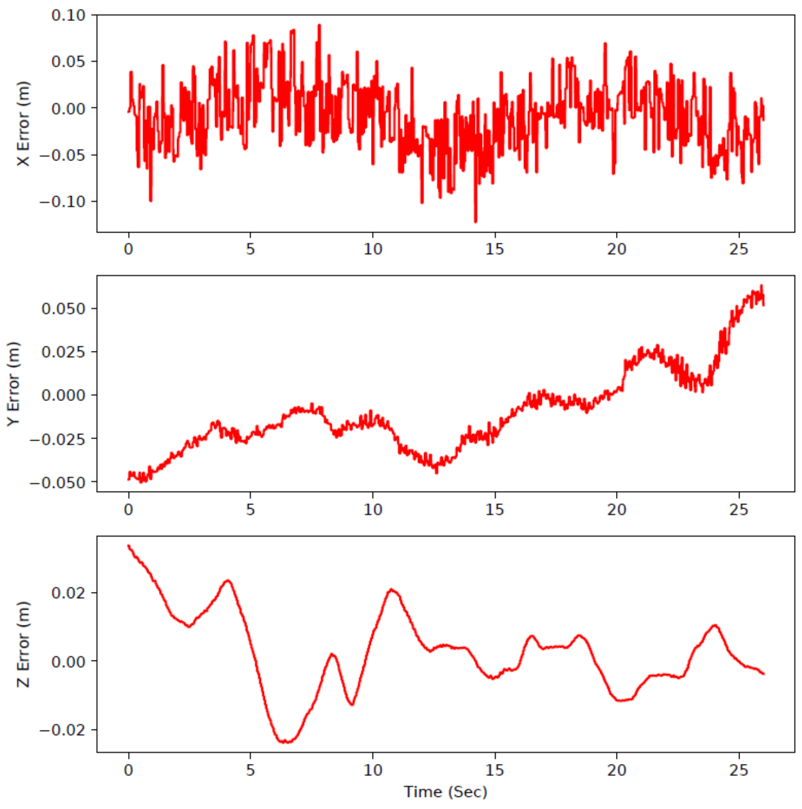


Figure 11. Control errors history during the whole aerial robotic arm moving.

The DCS was activated in this flight, moving the battery to compensate for the change of CoG generated by movements of the robotic arm. The standard deviation of control error in the world-fixed frame is about 3.64 cm in x axis, 2.37 cm in y axis and 1.16 cm in in z axis, which indicated our system can keep hovering at the certain point whether the robotic arm is moving or not. We noticed that during the whole test, the error increased at several points including 5, 10 and 22 s, respectively. This was caused by the moving of the robotic arm, but our system could always stabilize itself. Figure 12 shows some snapshots of the robotic arm aerial moving experiments.

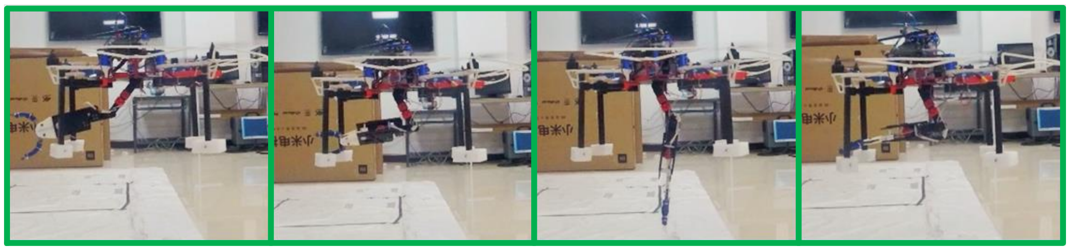
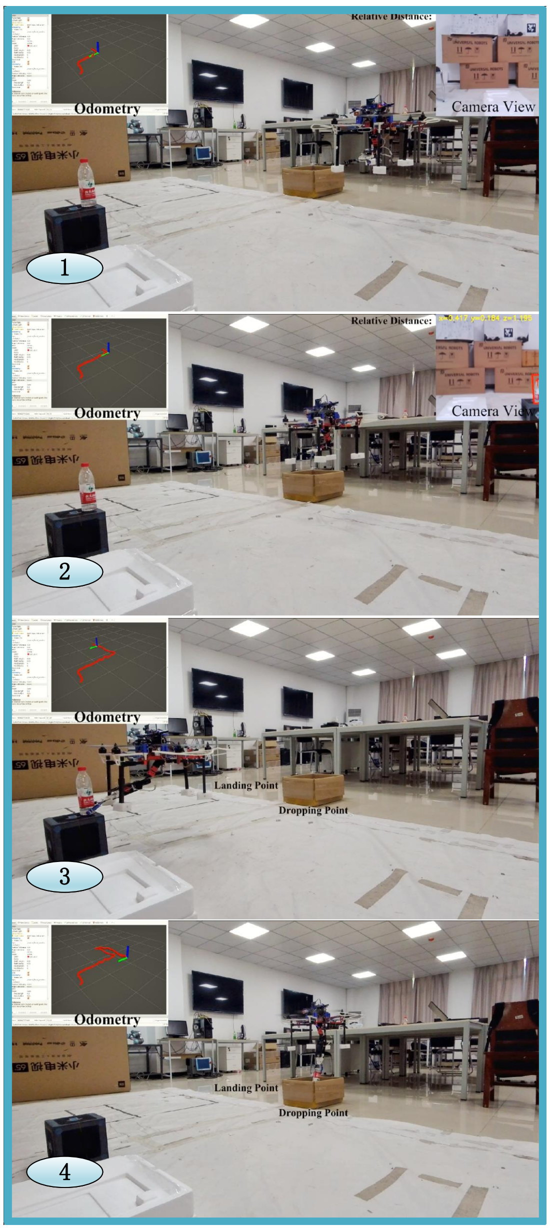
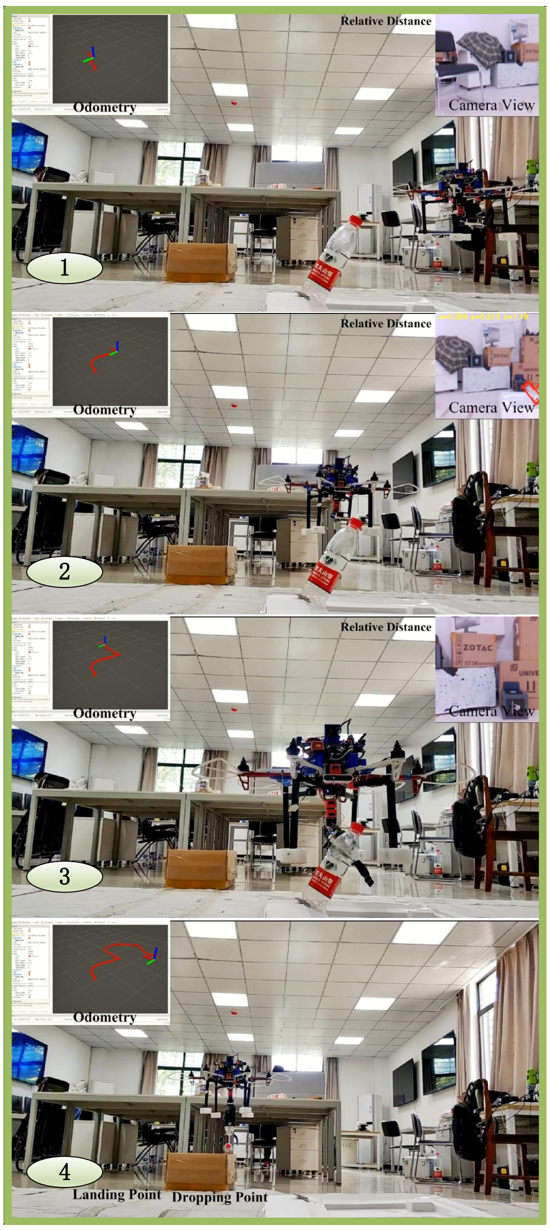


Figure 12. Snapshots of the robotic arm aerial moving experiments. Many movements are carried out during this flight to test the system. Each subfigure indicates a kind of movement of the robotic arm using in this experiment.

5.4. Autonomous Grasping

Autonomous grasping experiments with objects placed vertically and obliquely have also been conducted. Figure 13 presents the key steps of these two grasping experiments. The subfigures were taken at the specific moments ordered by the number in the bottom left corner: ① indicated the UAM is searching specific target. Here, we used the plastic bottle as the target; ② indicated the UAM has detected the target and given its relative position. The UAM will then align its position to the grasping position; ③ indicated the UAM hovering at the grasping position and the end effector is about to grasp the bottle. ④ indicated the UAM dropping the bottle at the dropping point. The odometry given by the onboard EKF in 100 hz rates were simultaneously presented in the top left corner, showing our system can run indoors without using any external visual motion caption system. The detection results and the calculated relative distance of the target objects were also simultaneously presented in the top right corner during searching.





(**a**) (**b**)

Figure 13. Snapshots of autonomous grasping experiments. The number in subfigures indicate selected moments: ① indicated the UAM is searching specific target; ② indicated the UAM detected the target and gives its relative position; ③ indicated the UAM hovering at the grasping position, and the end effector is about to grasp the bottle. ④ indicated the UAM dropping the grasped target at the dropping points. (**a**) Autonomous grasping of a bottle placed vertically; (**b**) autonomous grasping of a bottle placed obliquely.

In Figure 13a, the target was vertically placed at the top of a cube. By comparing images captured at ① and ②, it is easily to know the bottle is not shown in ① but detected in ②. The UAM was using a constant speeded to search in ①, and speeded up to get close to target after ②. The end-effector successfully approached and grabbed the target in ③ verified our methods worked well, and the plastic bottle was not flipped by the downward flow. In ④, the UAM can automatically drop the target at the dropping point, which indicates our system is capable of automatically finishing the whole grasping task.

In Figure 13b, the procedures were basically the same as mentioned above. What is different is that we placed the empty bottle with about 45° rotation. ② gave the detected target with its relative position and rotated angle. In ③, the end-effector used the rotation information of the target to successfully grasp the target.

These experimental results have shown that our system can accomplish an autonomous grasping mission.

6. Conclusions

In this paper, we developed an approach to enable a UAM to grasp oriented objects. The key challenges included detecting and locating objects in UAM perspective, CoG compensation, the flow influence to the objects, and indoor positioning of UAM system, all of which made the autonomous grasping mission very difficult. We showed the effectiveness of our approach in real tests with the ability to detect, locate, and grasp objects with arbitrary poses in GPS-denied environments without relying on the visual motion caption system. We believe that the proposed solution, both in terms of hardware and algorithms, will be useful in not only the aerial grasping missions but also in general grasping missions since the vision system can be applied to any kind of robotic arm. Future work will be set out to investigate how to estimate the 6D pose of objects in real-time. We will also improve the stability by using the servos with torque feedback and exploring a more advanced controller suit for the UAM.

Author Contributions: Conceptualization, S.L. and J.W.; methodology, S.L. and J.W.; software, S.L., J.W. and R.P.; validation, S.L., J.W. and R.P.; writing–original draft preparation, S.L. and J.W.; writing–review and editing, W.Y.; supervision, W.Y.; project administration, S.L. and W.Y.

Funding: The work was supported in part by the Funds for Creative Research Groups of Natural Science Foundation of Hubei Province under Grant 2018CFA006.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAM	Unmanned Aerial Manipulator
UAMs	Unmanned Aerial Manipulators
UAV	Unmanned Aerial Vehicle
UAVs	Unmanned Aerial Vehicles
2D	2-dimension
3D	3-dimension
SLAM	Simultaneous Localization and Mapping
YOLO	You Only Look Once
PLA	Poly Lactic Acid
ESC	Electronic Speed Controller
VPIB	Variable Parameter Integral Backstepping
IBVS	Image-Based Visual Servo
PID	Proportional Integral Derivative
ENU	East, North, and Up

D-H	Denavit-Hartenberg
CoG	Center of Gravity
ToF	Time of Flight
DoF	Degree of Freedom
ROS	Robot Operating Syste
VIO	Visual Inertial Odometry
EKF	Extended Kalman Filter
GPS	Global Positioning System
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
FPS	Frames Per Second
CFD	Computational Fluid Dynamics
IK	Inverse Kinematics
RRPN	Rotation Region Proposal Networks
RPN	Region Proposal Network
RoIs	Region of Interests
R-anchors	Rotation anchors
IoU	Intersection over Union
AP	Average Precision
NMS	Non-maximum Suppression
CPU	Central Processing Unit
GPU	Graphics Processing Unit

References

1. Lendzioch, T.; Langhammer, J.; Jenicek, M. Estimating Snow Depth and Leaf Area Index Based on UAV Digital Photogrammetry. *Sensors* **2019**, *19*, 1027, doi:10.3390/s19051027.
2. Olivares-Mendez, M.A.; Fu, C.; Ludivig, P.; Bissyandé, T.F.; Kannan, S.; Zurad, M.; Annaiyan, A.; Voos, H.; Campoy, P. Towards an Autonomous Vision-Based Unmanned Aerial System against Wildlife Poachers. *Sensors* **2015**, *15*, 31362–31391. doi:10.3390/s151229861.
3. Ruggiero, F.; Lippiello, V.; Ollero, A. Aerial Manipulation: A Literature Review. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1957–1964. doi:10.1109/LRA.2018.2808541.
4. Kim, S.; Choi, S.; Kim, H.J. Aerial manipulation using a quadrotor with a two DOF robotic arm. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 4990–4995. doi:10.1109/IROS.2013.6697077.
5. Korpela, C.; Orsag, M.; Danko, T.; Oh, P. Insertion tasks using an aerial manipulator. In Proceedings of the 2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, USA, 14–15 April 2014; pp. 1–6. doi:10.1109/TePRA.2014.6869148.
6. Božek, P.; Al Akkad M, A.; Blištan, P.; Ibrahim N, I. Navigation control and stability investigation of a mobile robot based on a hexacopter equipped with an integrated manipulator. *Int. J. Adv. Robot. Syst.* **2017**, *14*, 1–13.
7. Arleo, G.; Caccavale, F.; Muscio, G.; Pierri, F. Control of quadrotor aerial vehicles equipped with a robotic arm. In Proceedings of the 21st Mediterranean Conference on Control and Automation (MED), Chania, Greece, 25–28 June 2013; pp. 1174–1180.
8. Jimenez-Cano, A.; Martin, J.; Heredia, G.; Ollero, A.; Cano, R. Control of an aerial robot with multi-link arm for assembly tasks. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 4916–4921.
9. Ruggiero, F.; Trujillo, M.A.; Cano, R.; Ascorbe, H.; Viguria, A.; Peréz, C.; Lippiello, V.; Ollero, A.; Siciliano, B. A multilayer control for multirotor UAVs equipped with a servo robot arm. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 4014–4020.

10. Ohnishi, Y.; Takaki, T.; Aoyama, T.; Ishii, I. Development of a 4-joint 3-DOF robotic arm with anti-reaction force mechanism for a multicopter. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 985–991. doi:10.1109/IROS.2017.8202265.
11. Fabra, F.; Zamora, W.; Masanet, J.; Calafate, C.T.; Cano, J.C.; Manzoni, P. Automatic system supporting multicopter swarms with manual guidance. *Comput. Electr. Eng.* **2019**, *74*, 413–428.
12. Barka, E.; Kerrache, C.; Hussain, R.; Lagraa, N.; Lakas, A.; Bouk, S. A Trusted Lightweight Communication Strategy for Flying Named Data Networking. *Sensors* **2018**, *18*, 2683.
13. Kim, S.; Seo, H.; Choi, S.; Kim, H.J. Vision-Guided Aerial Manipulation Using a Multirotor With a Robotic Arm. *IEEE/ASME Trans. Mech.* **2016**, *21*, 1912–1923. doi:10.1109/TMECH.2016.2523602.
14. Ramon Soria, P.; Arrue, B.C.; Ollero, A. Detection, location and grasping objects using a stereo sensor on UAV in outdoor environments. *Sensors* **2017**, *17*, 103.
15. Kanellakis, C.; Terreran, M.; Kominiak, D.; Nikolakopoulos, G. On vision enabled aerial manipulation for multirotors. In Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017; pp. 1–7, doi:10.1109/ETFA.2017.8247653.
16. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 28: 29th Annual Conference on Neural Information Processing Systems 2015, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
17. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
18. Wu, B.; Wan, A.; Iandola, F.; Jin, P.H.; Keutzer, K. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
19. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv* **2016**, arXiv:1602.07360.
20. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
21. Ma, J.; Shao, W.; Ye, H.; Wang, L.; Wang, H.; Zheng, Y.; Xue, X. Arbitrary-Oriented Scene Text Detection via Rotation Proposals. *IEEE Trans. Multimed.* **2018**, *20*, 3111–3122, doi:10.1109/TMM.2018.2818020.
22. Yang, X.; Sun, H.; Fu, K.; Yang, J.; Sun, X.; Yan, M.; Guo, Z. Automatic ship detection in remote sensing images from google earth of complex scenes based on multiscale rotation dense feature pyramid networks. *Remote Sens.* **2018**, *10*, 132.
23. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020.
24. Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. *Robotics: Modelling, Planning And Control*; Springer: London, UK, 2010.
25. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 17 May 2009; Volume 3, p. 5.
26. Delmerico, J.; Scaramuzza, D. A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots. *Memory* **2018**, *10*, 20.
27. Rehder, J.; Nikolic, J.; Schneider, T.; Hinzmann, T.; Siegwart, R. Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4304–4311.
28. Diaz, P.V.; Yoony, S. High-Fidelity Computational Aerodynamics of Multi-Rotor Unmanned Aerial Vehicles. In Proceedings of the 2018 56th AIAA Aerospace Sciences Meeting, Kissimmee, FL, USA, 8–12 January 2018; p. 1266.

29. Tripathi, S.; Dane, G.; Kang, B.; Bhaskaran, V.; Nguyen, T. LCDet: Low-Complexity Fully-Convolutional Neural Networks for Object Detection in Embedded Systems. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 411–420, doi:10.1109/CVPRW.2017.56.
30. Xie, L.; Ahmad, T.; Jin, L.; Liu, Y.; Zhang, S. A New CNN-Based Method for Multi-Directional Car License Plate Detection. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 507–517.
31. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
32. Wang, J.; Guo, W.; Pan, T.; Yu, H.; Duan, L.; Yang, W. Bottle Detection in the Wild Using Low-Altitude Unmanned Aerial Vehicles. In Proceedings of the 2018 21st International Conference on Information Fusion (FUSION), Cambridge, UK, 10–13 July 2018; pp. 439–444, doi:10.23919/ICIF.2018.8455565.
33. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Savannah, GA, USA, 2–4 November 2016; Volume 16, pp. 265–283.