Toward Autonomous Rotation-Aware Unmanned Aerial Grasping

Shijie Lin[1*], Jinwang Wang[1*], Wen Yang[1] and Guisong Xia[2]

Abstract— **Autonomous Unmanned Aerial Manipulators (UAMs) have shown promising potentials to transform passive sensing missions into active 3-dimension interactive missions, but they still suffer from some difficulties impeding their wide applications, such as target detection and stabilization. This letter presents a vision-based autonomous UAM with a 3DoF robotic arm for rotational grasping, with a compensation on displacement for center of gravity. First, the hardware, software architecture and state estimation methods are detailed. All the mechanical designs are fully provided as open-source hardware for the reuse by the community. Then, we analyze the flow distribution generated by rotors and plan the robotic arm's motion based on this analysis. Next, a novel detection approach called Rotation-SqueezeDet is proposed to enable rotation-aware grasping, which can give the target position and rotation angle in near real-time on Jetson TX2. Finally, the effectiveness of the proposed scheme is validated in multiple experimental trials, highlighting it's applicability of autonomous aerial grasping in GPS-denied environments.**

I. INTRODUCTION

Unmanned aerial manipulators (UAMs) are known as one specific type of unmanned aerial vehicles (UAVs) equipped with one or multiple robotic arms and have attracted a lot research in recent years [1]. One main advantage of UAM is that it shows promising potentials to transform passive sensing missions into active 3-dimension (3D) interactive missions like grasping [2] and assembling [3]. The capabilities like aerial maneuvering and hovering make it possible for UAM to accomplish dangerous missions like grasping the rubbish on the cliff in scenic spots. Fig. 1(d) illustrates some dangerous and costly rubbish cleaning works.

For the autonomous aerial manipulating 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 robotic arm, the complex aerodynamics effects. Many efforts have been done to reduce these effects.

A comprehensive dynamic model of hexacopter with a robotic arm has been built in [4], analyzed the effects of CoG change and mass distributions. For successfully accomplishing the insertion task, a two-stage cascaded PID controller

*The first two authors contribute equally to this letter.

[1]Shijie Lin, Jinwang Wang and Wen Yang are with the School of Electronic Information, Wuhan University Wuhan 430072, China `linshijie@whu.edu.cn`; `jwwangchn@whu.edu.cn`; `yangwen@whu.edu.cn`.

[2]Guisong Xia are with the State Key Laboratory of Information Engineering, Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, Wuhan, 430079 China `guisong.xia@whu.edu.cn`.
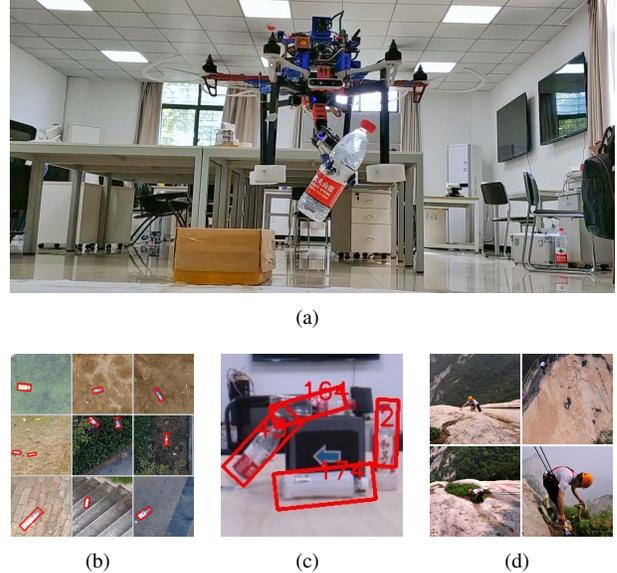
Fig. 1. (a) An unmanned aerial manipulator is grasping a plastic bottle to accomplish autonomous grasping tasks. (b) Sample images in UAV-DB dataset with plastic bottle groundtruth labeled by red box. (c) Detection results of Rotation-SqueezeDet with 2D box and rotation angle. (d) Workers are carrying out dangerous garbage disposal work on the cliff.

has been proposed in [3]. A Variable Parameter Integral Backstepping (VPIB) controller proposed in [5] can control an UAM with better performance than PID controller. With a movable compensation mechanism, a multilayer architecture controller compensate the internal and external effects layer by layer to control the UAM was presented [6]. To suppress the torque generated by the movements of robotic arm, a novel mechanism with a simplified model has been adopted in [7].

Usually, human operators are unable to accurately control the UAM due to observation change and data transmission delay. Such inaccuracy causes the robotic arm to be difficult to align and grasp the targets. Therefore it is better that the UAM can work without relying on external control. For fully autonomous grasping, the UAM needs perception ability. However, currently only a few works [8], [9], [10] considered robust visual perception aided autonomous grasping. An Image-Based Visual Servo (IBVS) was implemented in [8] to help locating the position of the targeted object. Feature models are used in [9] to find the position of known targets. In [8], correlation filters were adopted to track targets.

In recent years, End-to-End object detection algorithms like [11], [12] and [13] have shown surprising results. However, attempts directly applying these methods in the

UAM system usually end with failure or unsatisfactory robustness. The reason lies in the objects, like bottles, in UAV perspective often appeared with arbitrary poses as shown in Fig. 1(b), since the camera is tight-coupled and rotated with the UAV body. Moreover, most grasping missions need target's rotation angle for not touching the target when the end-effector is approaching. Furthermore, the performance like robustness and efficiency of the detection algorithms play an important role in grasping missions since it can bring more mobility to the UAM. However, even with NVIDIA Jetson TX2, current common oriented bounding boxes based methods like [14], [15] are still unable to run onboard since both require more than 11G GPU memory and Jetson TX2 only has 8G GPU memory.

In order to solve the problems mentioned above, we propose Rotation-SqueezeDet which can regress rotation angle and position in 2-dimension (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, 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) as descriptor 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 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 Fig. 1(c).

When applying the UAM into real world scenes, the flow generated by rotors can easily blew the lightweight targets away, like empty plastic bottles, and it is hard to predict whether an object can be easily blew away by the downward flow or not. Hence, a safety grasping action is better to be taken under weak or no flow influenced conditions. High-fidelity Computational Fluid Dynamics (CFD) simulation results of many different UAVs have been presented in [16]. From the observation of these results, the flow generated by each UAV's rotor is decreasing rapidly in the outter-wing area. Further, we roughly confirmed this observations from experiments by using an digital anemometer. To reduce the flow effects, the design and motion planning of robotic arm are in the light of the CFD sumulations and our flow measurements. Our experimental results have shown that grasping under weak or no flow influenced is possible.

The main contributions of this letter can be summarized as follows:
1) The Rotation-SqueezeDet method is proposed. This method can run on Jetson TX2 in near real-time and enable successful rotation-aware grasping. We believe that this method will be suitable in not only the aerial grasping but also general missions.
2) We consider the flow influence for designing and moving the robotic arm. The designed UAM can grasp lightweight objects in weak or no flow influenced area
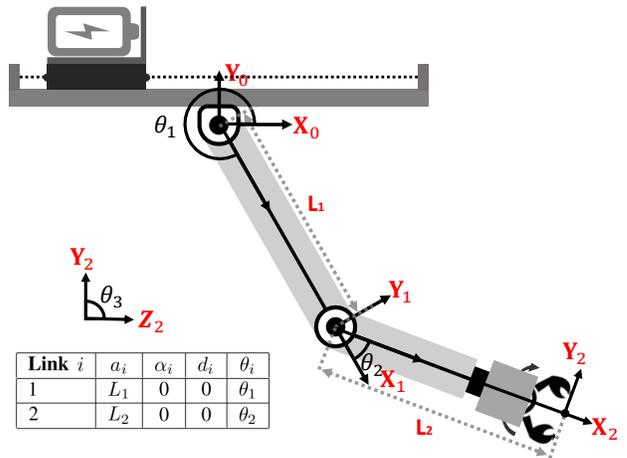

Fig. 2. Coornidates system and D-H parameter of the robotic arm.

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

during flight.
3) We have designed and assembled the UAM platform. The system is affordable since it costs less than $2300 USD and can fly without relying on expensive visual motion caption system. All mechanical structures are provided as open–hardware for reuse by the community. Link: `https://github.com/ele boss/UAMmech`

The rest of this letter is organized as follows. Section II describes the design and details of the overall system. Section III describes the motion planning and control of the robotic arm. Section IV describes the complete vision system including Rotation-SqueezeDet. Section V experimentally demonstrates the system including autonomous grasping and vision system performance. Finally, the conclusion and future work are presented in Section VI.

II. SYSTEM DESCRIPTION

A. Notation

The East, North and Up (ENU) coordinate system is used as world-fixed inertial frame corresponding to $\{x_w, y_w, z_w\}$. Following the definition of well known Denavit-Hartenberg (D-H) parameters [17], the link frame of Link i is defined as the $\{x_i, y_i, z_i\}$, $i = 0$ is the fixed arm frame. The definition of link frame is detailed in Fig. 2 and the table in the bottom left gives the D-H parameters of 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.

B. Hardware

In this work, a modified DJI hexacopter frame F550[1] is adopted. The hardware components are shown in Fig. 3.

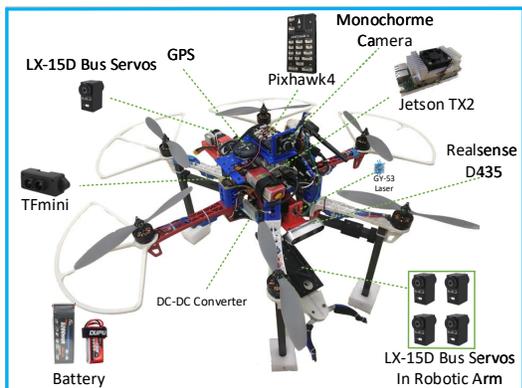[1] `www.dji.com/flame-wheel-arf/feature`

Fig. 3. Hardware components of the UAM

Fig. 4. An overview of the software architecture.

The propulsion module of the UAM is composed of Sunnysky[2] x3108s motor, Hobbywing[3] platinum Electronic Speed Controller (ESC), and 10 inches propeller. The Pixhawk4[4] autopilot with PX4[5] V1.8.0 flight stack and NVIDIA Jetson TX2[6] are placed at the top of the UAM as the main computing devices. A global shutter monochrome camera[7] is tight-coupled with Pixhawk4 by using a 3D-printed anti-vibration damping plate. A Benewake TFmini[8] laser rangefinder is chosen for being cheap, lightweight and with up to 12m maximum detection range, mounted downward facing to privide altitude feedback. An Intel RealSense D435[9] camera is mounted at the middle of the drone facing forward to find the targets.

We build a displacement compensation system (DCS) which can move counterweight to align the CoG and thus improves the stability of the total system. The DCS is mounted in the middle of UAM and made by 3D printed PLA material, including tow rails, a slide table and a bus servo to provide drive force. The LEBOT[10] LX-15D serial bus servos are 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 can only use one serial port in Jetson TX2 to control all servos. While the LX-15D servo can only provide $240°$ feedback, we use a short range time of flight (ToF) laser rangefinder GY-53 to provide the battery position feedback.

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

[2] www.rcsunnysky.com
[3] www.hobbywing.com
[4] www.holybro.com/product/55
[5] https://github.com/PX4/Firmware
[6] developer.nvidia.com/embedded/buy/jetson-tx2
[7] www.jinyandianzi.com
[8] www.benewake.com/tfmini.html
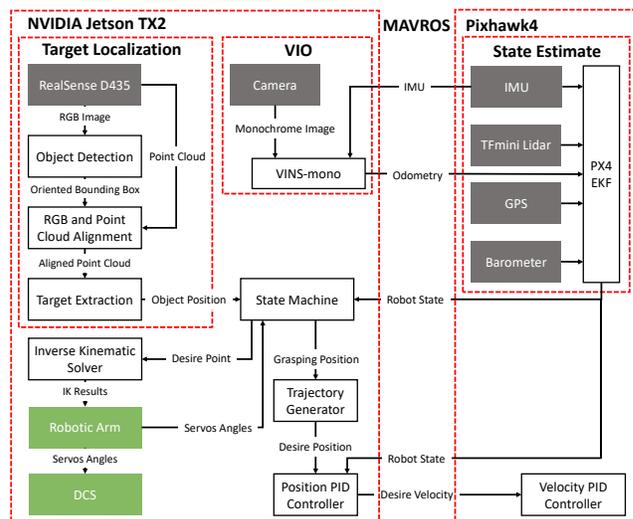[9] www.realsense.intel.com/stereo
[10] www.lobot-robot.com

Drive forces of robotic arm are also provided by LX-15D servos. The robotic arm has 3-DoF and can grasp objects by the end-effector. The first 2-DoF provide the robotic arm mobility to move at the planar 2D plane. The last DoF enables a rotational grasping by cooperating with vision system. The last DoF is of vital important for the rotated objects grasping, since the unrotated grasping can easily tip the object.

The total takeoff weight of the UAM is about 4.08kg. Thanks to the carbon fibre material and Poly Lactic Acid (PLA) material, the robotic arm is only weighted 0.459kg and has 43cm extended range.

C. Software Architecture

The software runs on two main processors: Pixhawk4 and Jetson TX2, and all processes are running onboard. Fig. 4 gives an overview of the system software architecture. Note that the Jetson TX2 is overclocked to run at max clock rate and unlocked 2 external CPU cores to generate more computing power.

Robot Operating System (ROS) [18] is a pseudo-operating system which allows developers to work cooperating by following its running mechanism. Modules related to state estimation and flying control are running on Pixhawk4 and exchange data with Jetson TX2 by MAVROS[11]. Visual Inertial Odometry (VIO) and object detection algorithms are running on Jetson TX2. A state machine is adopted to set the robotic arm motion and UAM waypoint. So the grasping position (x_w^s, y_w^s, z_w^s) of the UAM can be given by:

$$\begin{bmatrix} x_w^s \\ y_w^s \\ z_w^s \\ 1 \end{bmatrix} = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} + T_B^W T_C^B \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} - T_B^W T_0^B \begin{bmatrix} x_0^g \\ y_0^g \\ 0 \\ 1 \end{bmatrix} \quad (1)$$

where $T_B^W, T_0^B, T_C^B \in \mathbb{R}^{4 \times 4}$ are the homogeneous transformation matrix, T_0^B transforms the fixed arm frame to body

[11] https://github.com/mavlink/mavros

frame, T_C^B transforms the camera frame to the body frame, T_B^W transforms the body frame to the world-fixed frame, (x_0^g, y_0^g) is the grasping point.

We use the cascaded PID controller to control the drone, the position loop runs at TX2, the velocity loop runs at the Pixhawk4. The parameters of the cascaded PID are tuned when the robotic arm keeps static at the yellow star point (x_0^f, y_0^f) shown in Fig. 5.

D. 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.

Global Positioning System (GPS) usually fails to provide global positioning feedback when in indoor environment. Hence, in order to fly indoor, without using expensive visual motion caption system, we integrate the VINS-mono [19] VIO to provide the local position feedback, and it can provide the highest level of accuracy and robustness compared with multiple VIO [20]. 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 use the Inertial Measurement Unit (IMU) in Pixhawk4 as the inertial input, and a monochrome global shutter camera runs at 640×400 resolution and 90 Frames Pre Second (FPS) to provide clear images as visual input. To reduce drifting, the monochrome camera and Pixhawk4 are tight-coupled by using the 3D printed structures, the camera intrinsic matrix and camera to imu transformation parameters are carefully calibrated by using kalibr [21]. 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 30ms.

III. ROBOTIC ARM & DCS

A. 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, (x_0, y_0) is the
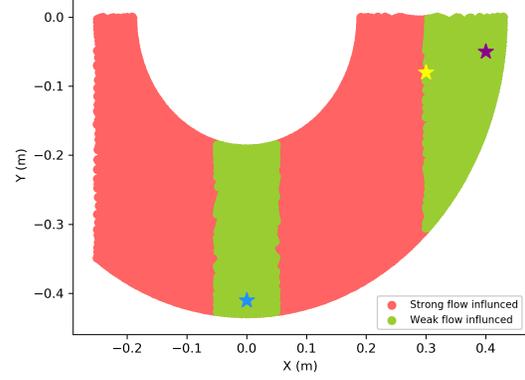

Fig. 5. Workspace of robotic arm printed by using forward kinematics considered flow influence.

point in arm fixed frame. Since the θ_3 is only related to target rotation angle θ, the workspace of our robotic arm is equal to a planar 2DoF robotic arm model. The table presented in Fig. 2 gives a clear D-H parameters definition of the robotic arm.

Based on the flow measurements which will be described in section V-B and mechanical limits, the actual workspace is given in Fig. 5, the green points indicate weak or no flow influenced area, 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 depend on the applications. We use Inverse Kinematics (IK) to solve the desired rotation angle of each joints. Assumed the robotic arm is planning to move to (x_0, y_0) point, the IK result is given by:

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

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

$$
\begin{aligned}
\theta_1 = \; &\arctan(x_0^2 / y_0^2) - \\
&\arccos(\frac{x_0^2 + y_0^2 + L_1^2 - L_2^2}{2L_1 \sqrt{x_0^2 + y_0^2}})
\end{aligned}
\tag{4}
$$

B. CoG Compensation

When the UAM is static on the ground and the robotic arm hold 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 adopte the strategy presented in [6] called DCS, moving the battery as a counter-weight since it's weight can provide sufficient compensation in relatively short moving distance.

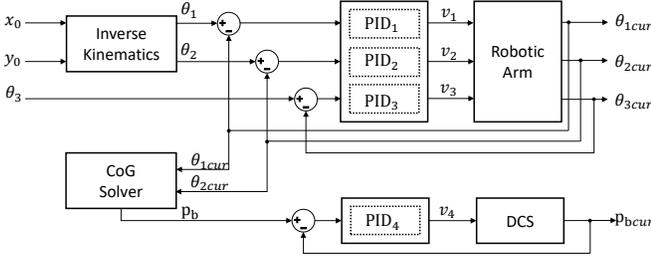CoG transformation of Link i and the end-effector payload

Fig. 6. Control diagram of the DCS and robotic arm.

Fig. 7. Rotation-SqueezeDet's detection pipeline.

from link frame to the body frame is 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 $T_i^0, T_0^B \in \mathbb{R}^{4 \times 4}$ are the homogeneous transformation matrices, T_0^B transforms from fixed arm frame to body frame, 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, 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 unstable. And this method works well in limited speed movement of robotic arm. To guarantee the compensation performance, we limit the rotation speed of each joint in robotic arm to make sure it does not exceed the maximum compensation speed of the linear slider.

C. Control

The total control diagram of the robotic arm and DCS are shown in Fig. 6. In Fig .6, v_1, v_2, v_3, v_4 is the rotation speed of the servos and the $\theta_{1cur}, \theta_{2cur}, \theta_{3cur}, p_{bcur}$ indicate the current feedback. Three linear PID controllers are adopted 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 a stable and smooth control.

IV. VISION SYSTEM

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

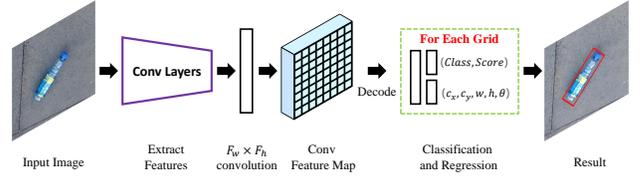Our proposed model is inspired by SqueezeDet [22] and Rotation Region Proposal Networks (RRPN) [14]. As the former, SqueezeDet is a single-pass detection pipeline combining bounding box localization and classification by a single network. It appears to be the smallest object detector by virtue of powerful but small backend network of SqueezeNet [13], [23]. As the latter, RRPN is based on Faster R-CNN [11], it 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 Region Proposal Network (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. However, in RRPN, instead, it uses Rotation anchors (R-anchors) and rotation RoI pooling, brings 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 [14]. 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 424×240 pixels image on Jetson TX2 and easy to train. However, SqueezeDet cannot predict the θ of 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, it can run on Jetson TX2 in near real time.

A. Network Architecture

The overall model of Rotation-SqueezeDet is illustrated in Fig. 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 conv 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. Notice that (c_x, c_y, w, h, θ) are decoded from predicted parameters tuple v, which is the direct outputs from the conv feature map. Here v are encoded

as follow:

$$v_x = \frac{c_x - c_x^a}{w^a}, v_y = \frac{c_y - c_y^a}{h^a},$$

$$v_w = \log \frac{w}{w^a}, v_h = \log \frac{h}{h^a}, v_\theta = \theta - \theta^a + k\pi \quad (7)$$

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 $\theta \in [0, \pi)$.

B. Oriented IoU Computation

To efficiently predict the rotation angle of object, we introduce a parallel 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 are poor because they cannot compute parallelly. Thus, we use a simple and efficient method to approximately compute the IoU parallelly, which is to use the angle deviation of two oriented bounding boxes. The approximate IoU [24] can be computed by:

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

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

C. 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 [25] 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.

D. 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 RGB 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:

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

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 ith 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
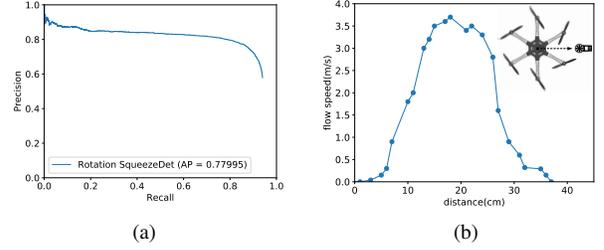

Fig. 8. (a) AP results of Rotation-SqueezeNet validated in UAV-BD. (b) The flow speed measured by digital anemometer and the subfigure in the top right corner is the moving path of the anemometer during measurement.

selected central subarea of target's point clouds, which can be calculated by:

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

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

V. EXPERIMENTS

The experimental setup is mentioned in Section II. During the flight tests, all data are logged onboard with no external data transmission. A high resolution video about the experiments are available here: http://youtu.be/v_h GzN8VIAU

A. Vision System Results

We trained and evaluated our vision system based on our pervious work UAV-BD [26], a bottle image dataset under UAV perspective. It contains about $34, 791$ object instances in $25, 407$ images labelled 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 [27]. We used the pertrained 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 Fig. 8(a), the AP of Rotation-SqueezeDet on UAV-BD is about 78.0% when $\text{IoU} = 0.5$. The run-time on Jetson TX2 is about 41ms with the size of 424×240 pixels color image.

B. Flow Distribution Validation

In order to verify the simulation results presented in [16] and give a rough parameter estimation for the planning mentioned in Section III. We disarmed the UAM on the ground and used Smart AS856[12] anemometer to measure

---

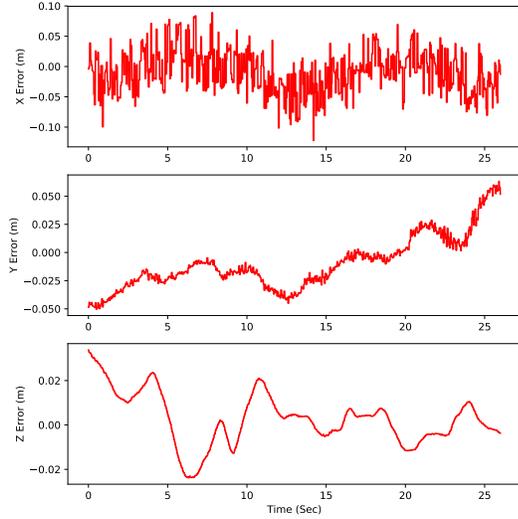[12] en.smartsensor.cn/products_detail/productId= 248.html

Fig. 9. Control errors history during the whole aerial robotic arm moving.

Fig. 10. Snapshots of the robotic arm aerial moving experiments. Many movements are carried out during this flight to test the system.

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 2D flow distribution to give a rough estimation for the robotic arm planning. The moving path of anemometer is given in the top right corner of Fig. 8(b) and the vertical distance from the anemometer to the rotor is about 16cm. Following this path, we recorded the average value of 5 measurements at one point and drew the curve in Fig. 8(b). Hence, the horizontal axis is the distance from the central position of UAM body to the outside following this path. From the observation of Fig. 8(b), the flow speed is decreasing rapidly at about 8cm and 30cm, and reaching the top speed at about 21cm which is the underside of rotors. The 8cm is close to inside of the UAM, the 30cm is close to the outside of the UAM. So the flow speed is relative weak when at the central position and outside position of the UAM, this observation basically agrees with the simulation results in [16].

C. 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 Fig. 9, the control errors of the system during this flight were logged and presented. After finishing takeoff procedure, the robotic arm will automatically move to multiple points and rotate the end-effector 90°. The DCS was activated in this flight, moving the battery to compensate the change of

CoG generated by movements of robotic arm. The standard deviation of control error in world-fixed frame are about 3.64cm in x axis, 2.37cm in y axis and 1.16cm in in z axis, indicated our system can keep hovering at the certain point no matter 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 seconds, respectively. This was caused by the moving of the robotic arm, but our system could always stabilize itself. Fig. 10 shows some snapshots of the robotic arm aerial moving experiments.

D. Autonomous Grasping

Autonomous grasping experiments with objects placed vertically and obliquely have also been conducted. Fig. 11 presents the key steps of these two grasping experiments. The subfigures were taken at specific moments ordered by the number in the bottom left corner: ① indicated the UAM is searching specific target. Here we use the plastic bottle as 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 100hz rates were simultaneously presented in the top left corner, showing our system can run indoor 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.
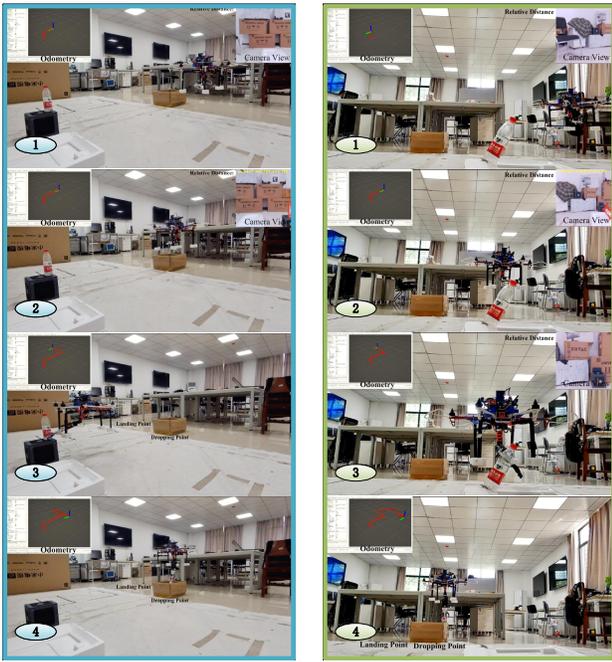
In Fig. 11(a), the target was vertically placed at the top of a cube which weight is about 120g. By comparing images captured at ① and ②, it is easily to know the bottle is not shown in ① but detected in ②. The UAM was using a constant speeded to search in ①, and speeded up to get close to target after 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 indicates our system is capable of automatically finishing the whole grasping task.

In Fig. 11(b), the procedures were basically same as mentioned above. What different is we placed the empty bottle with about 45° rotation by using tapes. 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 has capability to accomplish autonomous grasping mission.

VI. CONCLUSION AND FUTURE WORK

In this letter, we developed an approach to enable a UAM to grasp lightweight objects. The key challenges included detecting and locating objects in UAM perspective, CoG compensation, and the flow influence to the objects, all of which made the autonomous grasping mission very difficult. We showed the effectiveness of our approach in real tests

(a) Autonomous grasping of a bottle placed vertically

(b) Autonomous grasping of a bottle placed obliquely

Fig. 11. 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.

with the ability to detect, locate and grasp objects with arbitrary pose 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 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 object in real-time. We will also improve the stability by using the servos with torque feedback and adopting more advanced controller for the UAM.

REFERENCES

[1] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, July 2018.

[2] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two dof robotic arm," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 4990–4995.

[3] C. Korpela, M. Orsag, T. Danko, and P. Oh, "Insertion tasks using an aerial manipulator," in *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, April 2014, pp. 1–6.

[4] P. Božek, A. Al Akkad M, P. Blištan, and I. Ibrahim N, "Navigation control and stability investigation of a mobile robot based on a hexacopter equipped with an integrated manipulator," *International Journal of Advanced Robotic Systems*, vol. 14, pp. 1–13, 2017.

[5] A. Jimenez-Cano, J. Martin, G. Heredia, A. Ollero, and R. Cano, "Control of an aerial robot with multi-link arm for assembly tasks," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 4916–4921.

[6] F. Ruggiero, M. A. Trujillo, R. Cano, H. Ascorbe, A. Viguria, C. Peréz, V. Lippiello, A. Ollero, and B. Siciliano, "A multilayer control for multirotor uavs equipped with a servo robot arm," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4014–4020.

[7] Y. Ohnishi, T. Takaki, T. Aoyama, and I. Ishii, "Development of a 4-joint 3-dof robotic arm with anti-reaction force mechanism for a multicopter," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 985–991.

[8] S. Kim, H. Seo, S. Choi, and H. J. Kim, "Vision-guided aerial manipulation using a multirotor with a robotic arm," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 4, pp. 1912–1923, Aug 2016.

[9] P. Ramon Soria, B. C. Arrue, and A. Ollero, "Detection, location and grasping objects using a stereo sensor on uav in outdoor environments," *Sensors*, vol. 17, no. 1, p. 103, 2017.

[10] C. Kanellakis, M. Terreran, D. Kominiak, and G. Nikolakopoulos, "On vision enabled aerial manipulation for multirotors," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2017, pp. 1–7.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[13] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[14] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3111–3122, Nov 2018.

[15] X. Yang, H. Sun, K. Fu, J. Yang, X. Sun, M. Yan, and Z. Guo, "Automatic ship detection in remote sensing images from google earth of complex scenes based on multiscale rotation dense feature pyramid networks," *Remote Sensing*, vol. 10, no. 1, p. 132, 2018.

[16] P. V. Diaz and S. Yoony, "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," in *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 1266.

[17] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[19] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[20] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," *Memory*, vol. 10, p. 20, 2018.

[21] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4304–4311.

[22] B. Wu, A. Wan, F. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] S. Tripathi, G. Dane, B. Kang, V. Bhaskaran, and T. Nguyen, "Lcdet: Low-complexity fully-convolutional neural networks for object detection in embedded systems," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 411–420.

[24] L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang, "A new cnn-based method for multi-directional car license plate detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 507–517, 2018.

[25] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.

[26] J. Wang, W. Guo, T. Pan, H. Yu, L. Duan, and W. Yang, "Bottle detection in the wild using low-altitude unmanned aerial vehicles," in

2018 21st International Conference on Information Fusion (FUSION), July 2018, pp. 439–444.

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning." in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, vol. 16, 2016, pp. 265–283.