2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Quadrocopter Ball Juggling

Mark Müller, Sergei Lupashin and Raffaello D'Andrea

Abstract— **This paper presents a method allowing a quadrocopter with a rigidly attached racket to hit a ball towards a target. An algorithm is developed to generate an open loop trajectory guiding the vehicle to a predicted impact point – the prediction is done by integrating forward the current position and velocity estimates from a Kalman filter. By examining the ball and vehicle trajectories before and after impact, the system estimates the ball's drag coefficient, the racket's coefficient of restitution and an aiming bias. These estimates are then fed back into the system's aiming algorithm to improve future performance. The algorithms are implemented for three different experiments: a single quadrocopter returning balls thrown by a human; two quadrocopters co-operatively juggling a ball back-and-forth; and a single quadrocopter attempting to juggle a ball on its own. Performance is demonstrated in the Flying Machine Arena at the ETH Zurich.**

I. Introduction

In this paper we describe a system that allows a quadrocopter to hit a ping-pong ball towards a target using an attached racket. This enables a single quadrocopter to juggle a ball, multiple quadrocopters to hit a ball back-and-forth, or a human and quadrocopter to play together. The term "juggling" here is used in the same sense as in soccer, where one tries to keep the ball in the air for as long as possible. A useful way of visualising the problem setup is to think of a "flying racket" (see Fig. 1) hitting a ball.

Hitting a ball is a visually engaging problem, with which everyone is acquainted, and any casual bystander can immediately judge how successful a system is. This problem involves various aspects: deciding when and where to hit the ball, and to what target; analysing the dynamics of the ball flight, ball/racket impact and the dynamics of the quadrocopter; generating a trajectory moving the quadrocopter to a state which hits the ball as desired, while respecting the dynamics of the quadrocopter; and estimating the state of the ball accurately enough to allow for useful predictions.

Robotic juggling and ball sports are popular research topics, and are seen as challenging dexterous tasks. Examples include robotic table tennis, from simplified ping-pong [1] to teaching a robotic arm to return a table tennis ball [2]. Other interesting cases are human/robot volleyball [3]; robot basketball [4] and the RoboCup robotic soccer championship [5]. An example of juggling per se can be found in [6], with another interesting case being "blind" juggling [7].

Due to their agility, and mechanical simplicity, quadrocopters have become a popular subject of research. A few examples of challenging tasks executed by quadrocopters are: dancing [8]; balancing an inverted pendulum [9]; aggressive manoeuvres such as flight through windows and perching [10]; and cooperative load-carrying [11].

The problem of hitting a ball also yields opportunities for exploring learning and adaptation: presented here are strategies to identify the drag properties of the ball, the racket's coefficient of restitution, and a strategy to compensate for aiming errors, allowing the system to improve its performance over time. This provides a significant boost in performance, but only represents an initial step at system-wide learning. We believe that this system, and systems like it, provide strong motivation to experiment with automatic learning in semi-constrained, dynamic environments.

The paper is organised as follows: in Section II we derive equations to model quadrocopter flight, ball flight and ball/racket impact. In Section III we present algorithms to estimate the ball state and predict the ball's trajectory, estimate the racket's coefficient of restitution and estimate an aiming bias. Then an algorithm to generate a trajectory for the quadrocopter is given in Section IV, followed by a discussion on the system architecture and experimental setup in Section V. Results from experiments are presented in Section VI. We attempt to explain why the system fails on occasion in Section VII and conclude in Section VIII.

II. Dynamics

We model the quadrocopter with three inputs (refer to Fig. 2): the angular accelerations $\dot{q}$ and $\dot{p}$, taken respectively about the vehicle's $\mathbf{x}$ and $\mathbf{y}$ axes, and the mass-normalised collective thrust, f. The thrust points along the racket normal, $\mathbf{n}$. The attitude of the quadrocopter is expressed using the $\mathbf{z}$-$\mathbf{y}$-$\mathbf{x}$ Euler angles, rotating from the inertial frame to the

Fig. 1. A quadrocopter with attached badminton racket head. Three retro-reflective markers are attached to the vehicle, with which the vehicle pose can be determined (here, two are partially obscured by the racket). The ball, shown lying on the racket, is also wrapped in the retro-reflective tape to be visible to the motion capture system.

The authors are with the Institute for Dynamic Systems and Control, ETH Zurich, Sonneggstrasse 3, 8092 Zurich, Switzerland.
{mullerm, sergeil, rdandrea}@ethz.ch

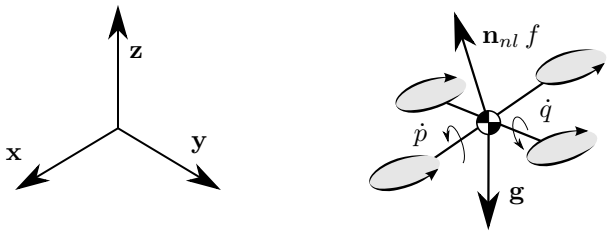

Fig. 2. Dynamics of a quadrocopter, acting under inputs of thrust (f), and two angular accelerations $\dot{q}$ and $\dot{p}$; and under the influence of gravity $\mathbf{g}$. The base vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ of the inertial reference frame are also shown.

body-fixed frame by yaw (ψ) first, then pitch (θ) and finally by roll (ϕ) – for simplicity, throughout this paper the yaw is assumed to be controlled to zero by a separate controller. Using the two remaining angles, we can express the racket normal in the inertial frame as

$$\mathbf{n}_{nl} = (\sin\theta\cos\phi,\ -\sin\phi,\ \cos\theta\cos\phi)\,. \tag{1}$$

In our analysis we assume that the pitch and roll angles remain small, and obtain $\sin\theta \approx \theta$ and $\cos\theta \approx 1$ to first order (likewise for ϕ). Furthermore, under this assumption, we can take the Euler angle accelerations $\ddot{\theta}$ and $\ddot{\phi}$ as equal to the vehicle's angular accelerations about its $\mathbf{x}$ and $\mathbf{y}$ axes.

The equation of motion for the quadrocopter is

$$\ddot{\mathbf{s}}_r = \mathbf{n}f + \mathbf{g} \tag{2}$$

$$\ddot{\theta} = \dot{q} \tag{3}$$

$$\ddot{\phi} = \dot{p} \tag{4}$$

where

$$\mathbf{n} = (\theta,\ -\phi,\ 1)\,,\quad \mathbf{g} = (0,\ 0,\ -g) \tag{5}$$

and $\mathbf{s}_r$ to denotes the quadrocopter's position in the inertial reference frame, and $\dot{\mathbf{s}}_r$ and $\ddot{\mathbf{s}}_r$ its velocity and acceleration, respectively.

The ball's flight is modelled as a point mass under the influence of gravity and aerodynamic drag, and we ignore spin. For a more detailed treatment of table tennis ball flight and impact dynamics, see [12]. We model drag as proportional to the ball's speed squared, with proportionality coefficient K_D. Denoting the ball's position in space as $\mathbf{s}_b$, we can write the ball's equation of motion as

$$\ddot{\mathbf{s}}_b = \mathbf{g} - K_D\|\dot{\mathbf{s}}_b\|\dot{\mathbf{s}}_b \tag{6}$$

where $\|\cdot\|$ refers to the Euclidean norm.

The impact between the racket and the ball is modelled as an impulse acting in the direction of the racket normal. The "efficiency" of the impact is captured by the coefficient of restitution, $\beta \in [0, 1]$. We define β as the ratio of the components of the ball's pre- and post-impact velocities ($\dot{\mathbf{s}}_b^-$ and $\dot{\mathbf{s}}_b^+$, respectively) in the direction of the racket normal, $\mathbf{n}$,

$$\beta = -\frac{\left(\dot{\mathbf{s}}_b^+\right)^T\mathbf{n}}{\left(\dot{\mathbf{s}}_b^-\right)^T\mathbf{n}} \quad \text{for} \quad \dot{\mathbf{s}}_r = \mathbf{0}, \tag{7}$$

derived for simplicity for the case of a fixed racket.

We assume that the mass of the vehicle/racket is much larger than that of the ball (meaning that the racket velocity remains unaffected by the impact) and that the contribution of the racket's angular velocity to the post-impact ball state is negligible. This is because we intend to hit the ball at the racket's centre, which approximately corresponds to the vehicle's centre of rotation. Then the ball's velocity after impact with a moving racket is

$$\dot{\mathbf{s}}_b^+ = \dot{\mathbf{s}}_b^- - (1+\beta)\left(\left(\dot{\mathbf{s}}_b^- - \dot{\mathbf{s}}_r\right)^T\mathbf{n}\right)\mathbf{n}. \tag{8}$$

Experiments showed that the ball's velocity tangential to the racket face does change during impact, but this change is very hard to predict and is therefore neglected.

III. Estimation

In this section we describe a Kalman filter to estimate the ball's state, and strategies for estimating the ball's drag coefficient, the coefficient of restitution of the racket, and an aiming bias. The values for coefficient of restitution and drag are estimated online, rather than measured and stored, since individual balls show large differences in behaviour. These estimates are used for prediction and aiming of the ball as used in Section IV.

A. Ball state

We define the ball state as

$$\mathbf{x}_b = (\mathbf{s}_b,\ \dot{\mathbf{s}}_b)\,. \tag{9}$$

If we ignore drag, we can write the evolution of the ball state during free flight as a continuous time linear system:

$$\dot{\mathbf{x}}_b = \mathbf{A}_c\mathbf{x}_b + \mathbf{B}_c, \tag{10}$$

where

$$\mathbf{A}_c = \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix},\ \mathbf{B}_c = \begin{bmatrix}\mathbf{0}_{3\times1}\\ \mathbf{g}\end{bmatrix}. \tag{11}$$

This we can convert to a discrete time system with step size τ_k, where $\mathbf{x}_b[k] := \mathbf{x}_b(t_k)$ is the state at time t_k. The process noise $\mathbf{w}[k]$ is modelled as acting only on the velocity components to capture unmodelled accelerations. The measurement noise $\mathbf{v}[k]$ is assumed to act equally in all directions upon the measurement, $\mathbf{z}[k]$. Then

$$\mathbf{x}_b[k+1] = \mathbf{A}[k]\mathbf{x}_b[k] + \mathbf{B}[k] + \mathbf{w}[k] \tag{12}$$

$$\mathbf{z}[k] = \mathbf{H}[k]\mathbf{x}_b[k] + \mathbf{v}[k], \tag{13}$$

where

$$\mathbf{A}[k] = \begin{bmatrix}\mathbf{I}_{3\times3} & \tau_k\,\mathbf{I}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\end{bmatrix} \tag{14}$$

$$\mathbf{B}[k] = \begin{bmatrix}0 & 0 & -\frac{1}{2}g\tau_k^2 & 0 & 0 & -g\tau_k\end{bmatrix}^T \tag{15}$$

$$\mathbf{H}[k] = \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix}. \tag{16}$$

An estimate of the ball state ($\hat{\mathbf{x}}_b[k]$) can now be calculated using a discrete time Kalman filter [13]. We modify the standard linear Kalman filter formulation by including drag in the state prediction step. We introduce an intermediate position

and velocity for integration, $\mathbf{r}(t)$ and $\dot{\mathbf{r}}(t)$, respectively. By using the estimate at time t_k as initial value, $\mathbf{r}$ and $\dot{\mathbf{r}}$ can be evaluated by numerical integration, so that

$$\left(\mathbf{r}\left(t_k\right), \dot{\mathbf{r}}\left(t_k\right)\right) := \hat{\mathbf{x}}_b[k] \tag{17}$$

and $\mathbf{r}(t)$, $\dot{\mathbf{r}}(t)$ satisfy (6).

The state prediction for the Kalman filter is then simply $\mathbf{r}(t_k + \tau_k)$ and $\dot{\mathbf{r}}(t_k + \tau_k)$. For the variance propagation, and for the measurement update of both the state and the variances, the usual linear Kalman filter equations are used.

Taking the time derivative of the ball's specific mechanical energy $E_b = \frac{1}{2}\|\dot{\mathbf{s}}_b\|^2 + \mathbf{g}^T\mathbf{s}_b$, yields the specific power extracted from the system by the drag force,

$$\dot{E}_b = -K_D\|\dot{\mathbf{s}}_b\|^3. \tag{18}$$

Using the ball state estimate $\hat{\mathbf{x}}_b[k]$ from the ball state filter to calculate the ball's speed $\bar{v}[k]$, the above can be approximated as a measurement $\tilde{K}_D[k]$,

$$\bar{v}[k] = \sqrt{\hat{\mathbf{x}}_b[k]^T \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3}\end{bmatrix} \hat{\mathbf{x}}_b[k]} \tag{19}$$

$$\bar{E}[k] = \frac{1}{2}\left(\bar{v}[k]\right)^2 + \begin{bmatrix}\mathbf{g}^T & \mathbf{0}_{1\times3}\end{bmatrix} \hat{\mathbf{x}}_b[k] \tag{20}$$

$$\tilde{K}_D[k] = -\frac{\left(\bar{E}[k] - \bar{E}[k-1]\right)}{\tau_k\left(\frac{1}{2}\left(\bar{v}[k] + \bar{v}[k-1]\right)\right)^3}, \tag{21}$$

where $\bar{E}[k]$ is the estimate of the ball's mechanical energy at time t_k, and we use a two step average of the ball's speed.

These measurements can now be used to form an estimate of the drag value ($\hat{K}_D[k]$) using a recursive least squares (RLS) estimator [14]. We define $P_D[k] = \text{var}\left(\hat{K}_D[k]\right)$ as the estimate variance, and $R_D[k]$ as the measurement noise variance (from a nominal variance $R_{D,0}$, weighted by a two-step average of the ball's speed). We also introduce an intermediary gain $C_D[k]$.

$$R_D[k] = \frac{R_{D,0}}{\left(\frac{1}{2}\left(\bar{v}[k] + \bar{v}[k-1]\right)\right)^3} \tag{22}$$

$$C_D[k] = \frac{P_D[k-1]}{P_D[k-1] + R_D[k]} \tag{23}$$

$$\hat{K}_D[k] = \hat{K}_D[k-1] + C_D[k]\left(\tilde{K}_D[k] - \hat{K}_D[k-1]\right) \tag{24}$$

$$P_D[k] = \frac{P_D[k-1]R_D[k]}{P_D[k-1] + R_D[k]} \tag{25}$$

We disallow measurements below some minimum height to ensure we only use measurements taken when the ball is in flight (i.e. not being handled, or hit by the racket). To protect against numerical issues, we disallow measurements when the ball's speed approaches zero.

For longer term predictions of the ball state, we numerically integrate forward the ball state using (6). This is used to predict the time of impact and the ball state at impact, as well as during aiming, to evaluate possible post-impact ball velocities. For such predictions an accurate estimate of the drag coefficient is crucial.

B. Racket coefficient of restitution

By examining the ball and vehicle state estimates before and after an impact, we can estimate the racket's coefficient of restitution using (8). These estimates can be combined to form an estimate $\hat{\beta}$, again using an RLS estimator, similar to (22) - (25), but with the measurement noise variance taken as constant.

We only allow coefficient of restitution measurements lying in the range $[0.5, 1]$. The lower bound allows us to identify "failed" impacts, for example if the ball hits the racket frame, or the propellers. The value of 0.5 was chosen by examining experimental data.

C. Aiming bias

We notice that with the preceding, the system still shows an aiming bias, defined as the difference between the target position and where the ball crossed the target height. The system attempts to identify where it should aim so that the ball hits the target point, by using two separate RLS estimators, one each for the $\mathbf{x}$ and $\mathbf{y}$ components of the aiming bias. This allows us to compensate for systematic errors, such as a misaligned racket.

IV. Trajectory generation

Here we derive an open-loop trajectory to move the quadrocopter from some initial state to an impact state, in the time remaining until impact. We define impact as when the ball drops through some user-defined impact height. From the ball estimator, we have an estimate of time remaining until impact (T), and the ball's velocity at this time.

To calculate the required ball velocity after impact, we note that the post-impact trajectory has to start at the impact point, pass through some maximum height, and reach the aiming point. These requirements define a unique ball trajectory. We note that, by (6), the ball always moves in a vertical plane, implying that we need only solve for horizontal and vertical components of the post-impact ball velocity (v_h and v_v, respectively).

We can solve for these using a two-dimensional gradient descent search, minimising the cost function $C\left(v_h, v_v\right)$, composed of a height error and a lateral distance error. We denote the achieved maximum height by $h_m(v_h, v_v)$, and the desired maximum ball height by $h_{m,d}$. Furthermore, we use the distance $l\left(v_h, v_v\right)$ at which the ball passes through the target height, and the desired distance l_d.

$$\begin{aligned} C\left(v_h, v_v\right) = &\left(h_m\left(v_h, v_v\right) - h_{m,d}\right)^2 \\ &+ \left(l\left(v_h, v_v\right) - l_d\right)^2 \end{aligned} \tag{26}$$

The functions $h_m\left(v_h, v_v\right)$ and $l\left(v_h, v_v\right)$ are evaluated by examining the trajectories resulting from numerical integration of (6).

Using the ball's pre- and post-impact velocities, we can solve for the required racket state at impact using (8), yielding

$$\mathbf{n}_{des}(T) = \frac{\dot{\mathbf{s}}_b^- - \dot{\mathbf{s}}_b^+}{\|\dot{\mathbf{s}}_b^- - \dot{\mathbf{s}}_b^+\|} \tag{27}$$

$$V_{\perp,des} := (\dot{\mathbf{s}}_r(T))^T \mathbf{n} = \frac{1}{1+\beta}(\beta \dot{\mathbf{s}}_b^- + \dot{\mathbf{s}}_b^+)^T \mathbf{n}, \qquad (28)$$

where $V_{\perp,des}$ is the desired racket speed in the direction of the desired racket normal ($\mathbf{n}_{des}$) at impact. Because of how we mount the racket (see Fig. 1), it follows that the racket state is simply that of the quadrocopter, displaced in the body $\mathbf{z}$-axis by some offset.

There are a total of six constraints we need to meet at impact. The desired pitch and roll angles can be derived from the desired racket normal at impact ($\mathbf{n}_{des}(T)$) using (1). We also need to achieve the normal speed $V_{\perp,des}$, and the quadrocopter has to be at the impact point, at time T.

Using (2) to describe the quadrocopter dynamics under the small angle assumption, we have three inputs: the mass-normalised thrust f, and the angular accelerations $\dot{q}$ and $\dot{p}$. Noting that we need to solve for six equations, we assume affine inputs of the form

$$f(t) = A_f t + B_f \qquad (29)$$
$$\dot{q}(t) = A_\theta t + B_\theta \qquad (30)$$
$$\dot{p}(t) = A_\phi t + B_\phi. \qquad (31)$$

This form is inspired by the analysis of a linear one-dimensional system where the ball and racket are constrained along the $\mathbf{z}$ axis, with the single control input being the racket's acceleration f. An affine input minimises the mechanical energy expended by the control effort. (29)-(31) are simple enough to yield closed-form solution, while providing the necessary degrees of freedom. By substituting these into (2), and integrating, we can write the quadrocopter's position and velocity as polynomials in time. We substitute for the six requirements at impact, and substitute the integration constants with the initial conditions, to yield six equations in six unknowns (the input coefficients). This system of equations can be reduced to a single fourth order equation in one unknown, which can be solved for in closed form (the full derivation is made available online on the first author's website).

For a real polynomial of order four, we can have either no, two, or four real solutions. In case of multiple real solutions, we note that we wish to avoid large inputs to not violate the small-angle assumption, and select a solution minimising:

$$A_f^2 + A_\theta^2 + A_\phi^2. \qquad (32)$$

Because of the simple relationships between the related affine constants, small values for A_f, A_θ and A_ϕ imply small B_f, B_θ and B_ϕ, respectively (refer to the online derivation).

We have now solved for open-loop inputs which move the system from some initial state to the state needed to hit the ball, in the time until impact. This trajectory will satisfy the quadrocopter equation of motion under the small-angle assumption, but it does not take actuator saturation into account. By saturation we mean that the individual motor commands will exceed what is possible, for example when the collective thrust exceeds the vehicle's achievable limits, typically at the beginning or end of the trajectory.

This open-loop trajectory is sent to a near-hover feedback controller, similar to that described in [15]. The controller runs on position feedback, and we use the calculated desired velocity, acceleration and body rates as feed-forward terms, to generate the four individual motor commands.

V. Experimental setup

The preceding were coded in C++, and implemented in the Flying Machine Arena (FMA) at the ETH Zurich. The algorithm is shown schematically in Fig. 3, and can be broken into the following components:

- The *ball estimator* is responsible for finding the ball in the space, estimating its state, and then predicting when and where impact will occur.
- The *trajectory generator* calculates the desired quadrocopter state at impact, and generates a set of inputs to move the vehicle to that state at impact time.
- *Impact identification* allows the system to use information from previous impacts to improve the system performance.
- The generated trajectory, and commands, are now sent to the *feedback controller*.

It is important to note here that the trajectory generator is invoked in two places – when a new trajectory is initialised, and as information about the impact becomes available. At initialisation, an initial state for the trajectory is fixed, which remains unchanged until the next impact. As the impact prediction improves, the desired end state of the quadrocopter is updated, and the equations are solved to generate a trajectory from the fixed initial condition, to the updated impact condition.

The resulting trajectory works well when moving over small lateral distances, but larger lateral distances imply larger angular deviations, violating our assumption of small θ and ϕ. Therefore, we use two slightly different implementations of the trajectory generator, differing in how the initial state is chosen. In the "continuous" case, we use the quadrocopter's state at the start of the manoeuvre,

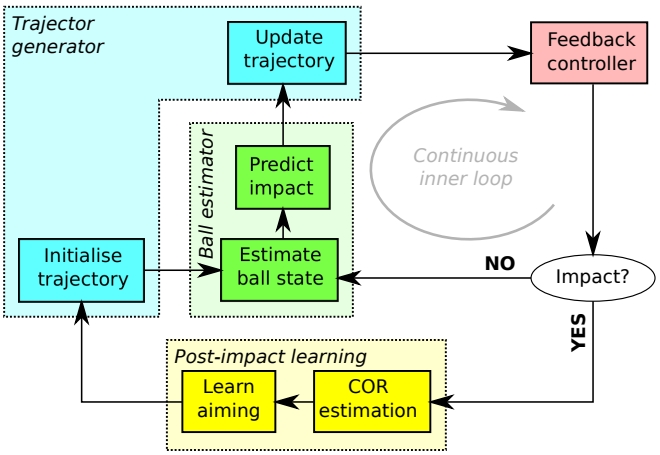

Fig. 3. Algorithm layout: the inner loop is executed continuously as new information becomes available, and the outer loop is only executed once per impact.

meaning that nominally the commands to the vehicle are continuous. This mode works best when moving over small lateral distances. The second, "discontinuous", mode is used when we know that we have more time until intercept, and works better when moving over larger distances. In this case we generate a nominal trajectory, starting some pre-defined distance below the predicted impact point, at zero velocity. In this mode, since the desired trajectory does not start from the quadrocopter's true initial state, the bulk of the initial control effort is due to the feedback controller bringing the vehicle onto the nominal path. The assumption is that by the time of impact, the vehicle will be on the desired trajectory, flying primarily on feed-forward commands.

The continuous mode is used for single quadrocopter juggling, while the discontinuous mode is used in the cases of multiple quadrocopter play, or quadrocopter-human play.

The FMA is a platform for design and validation of autonomous aerial systems and consists of a large ($10\,\mathrm{m} \times 10\,\mathrm{m} \times 10\,\mathrm{m}$) motion capture volume and a fleet of quadrocopters. The vehicles are sent commands for the three body rates and the collective thrust at $67\,\mathrm{Hz}$. An on-board controller uses rate gyro measurements to generate motor thrust commands at $800\,\mathrm{Hz}$. The vehicles are capable of following angular rate commands of up to $1900\,^\circ/\mathrm{s}$. More details about the Flying Machine Arena can be found in [8], [9] and [16].

Each vehicle is equipped with three retro-reflective motion capture markers; this allows the motion capture system to calculate the 6DOF vehicle pose for each frame, provided at $200\,\mathrm{Hz}$. Any markers not associated with vehicles are treated as point objects – the ping-pong ball is tracked as one such object. The FMA is equipped with 8 cameras; only 3 are required to see a marker to compute its position, providing a high degree of tracking redundancy even when markers are partially occluded, e.g. by a racket.

We use standard $40\,\mathrm{mm}$ diameter table tennis balls, wrapped in retro-reflective tape to be visible to the motion capture system. The ball is hit using a badminton racket head rigidly mounted on the vehicles – as shown in Fig. 1, the racket is mounted by placing the centre above the quadrocopter centre of mass, and aligning the racket normal with the quadrocopter **z**-axis. This minimises the effects of the angular velocity of the quadrocopter, and is mechanically convenient.

VI. Experimental results

Here we present results from three experiments. In each case the quadrocopter is attempting to hit the ball at a given intercept height, and maintain a specified maximum ball height. This maximum height was chosen by experiment as $2\,\mathrm{m}$ above the impact point.

1) **Returning a throw**: a single quadrocopter attempts return a thrown ball. This was used to demonstrate the effects of parameter identification.
2) **Cooperative juggling**: two quadrocopters attempt to hit a ball to one another – each quadrocopter has as target the other's starting position.

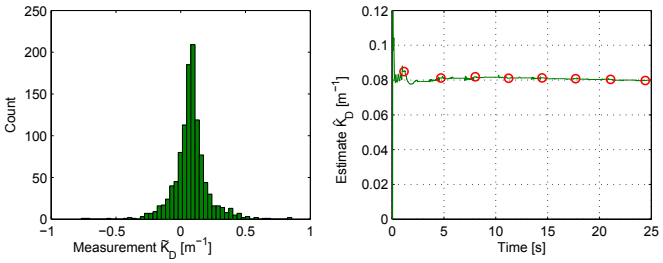

Fig. 4. Drag estimator outputs while a vehicle returns throws, showing a histogram of the individual drag measurements, and the evolution of the drag estimate. The measurements have as mean $0.078\,\mathrm{m}^{-1}$, and standard deviation $0.14\,\mathrm{m}^{-1}$. The red circles on the estimate evolution plot indicate when the ball was hit by the vehicle.

3) **Solo juggling**: a quadrocopter attempts to juggle a ball on its own, and keep the impact location fixed.

A video demonstrating these experiments is available online on the first author's website.

A. Returning a throw

1) Drag estimation: Fig. 4 shows that the drag estimate quickly converges to a value of approximately $0.079\,\mathrm{m}^{-1}$, and that the estimate is unaffected by the user handling the ball, quadrocopter impacts and impact with the ground.

As validation, we can estimate the aerodynamic drag using the usual $F_D = \frac{1}{2}\rho\|\dot{\mathbf{s}}_b\|^2 C_D A_{ball} = m_{ball} K_D \|\dot{\mathbf{s}}_b\|^2$, from which we have

$$K_D = \frac{\rho A_{ball} C_D}{2 m_{ball}}. \tag{33}$$

Taking $\rho = 1.2\,\mathrm{kg/m}^3$, and $C_D = 0.4$ [17], $A_{ball} = \pi r^2 = 1.257 \times 10^{-3}\,\mathrm{m}^2$ and a mass of $m_{ball} = 5 \times 10^{-3}\,\mathrm{kg}$, we get $K_{D_{calc}} = 0.06\,\mathrm{m}^{-1}$. This is in close agreement the estimate shown in Fig. 4.

2) Racket estimation: The racket estimator generates estimates of the racket's coefficient of restitution $\hat{\beta}$. Fig. 5 shows the coefficient of restitution estimate settling at a value of $\hat{\beta} \approx 0.76$. Interesting to note is how the measurements vary with position on the racket, showing a sweet spot near the centre of the racket.

The distribution of the impact points on the racket face are an indication of the system's ability to predict the impact

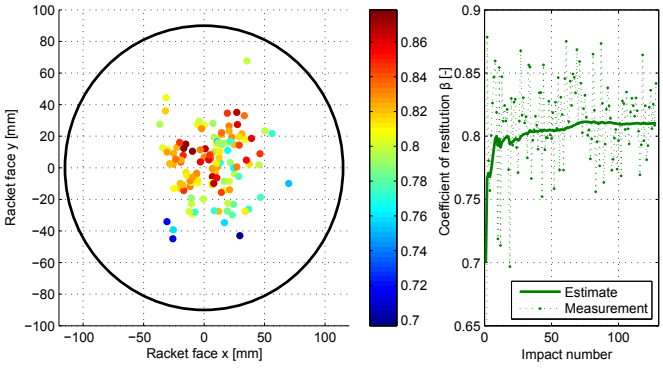

Fig. 5. Racket estimator output while a vehicle returns throws, showing coefficient of restitution measurements and the distribution of impacts on the racket face; and how the estimate evolved in time.

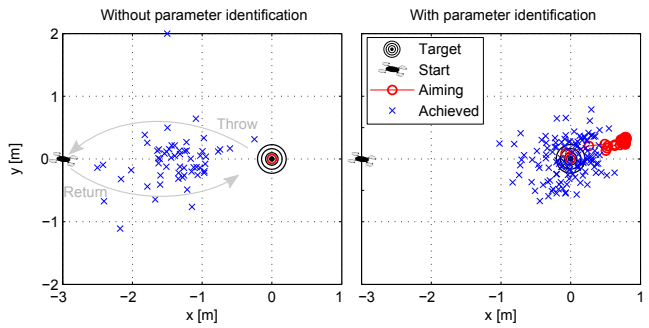

Fig. 6. Influence of parameter identification, for a single vehicle returning throws. On the left are the results when all identification is disabled, while the right shows the performance when starting with the same values and attempting to identify the parameters for drag, coefficient of restitution and aiming. The ball is thrown from the target position, and the quadrocopter attempts to return it – the quadrocopter starts at $x = -3\,\mathrm{m}$, $y = 0\,\mathrm{m}$.

point and time, and steer the vehicle to this impact point at the impact time. The reference frame used here is such that the racket face $\mathbf{x}$ and $\mathbf{y}$ align with the inertial $\mathbf{x}$ and $\mathbf{y}$, respectively, at zero pitch and roll angles.

3) Aiming: The aiming algorithm compares the point at which the ball actually lands to the point the quadrocopter wanted to hit, and attempts to shift the aiming point such that the ball lands on the target after the next impact.

The results of the aiming estimator are shown in Fig. 6, where we compare the system performance with and without parameter identification. I.e. on the right the system identifies the ball's drag characteristics, the racket's coefficient of restitution, and the aiming bias. The aiming bias is estimated at $760\,\mathrm{mm}$ in $\mathbf{x}$ and $270\,\mathrm{mm}$ in $\mathbf{y}$.

The mean error is reduced from $1.4\,\mathrm{m}$ without identification (for 58 hits), to a mean (over 15 hits) of $47\,\mathrm{mm}$ after the estimates have settled. We also notice that the standard deviation reduces slightly with identification – this is likely due to the way the experiment is conducted: if the ball is returned well, the throws will all start from similar locations. However, if the returns are poor and the user has to move to pick up the ball, we can expect a greater variation in the throws, and the resulting returns will show a higher standard deviation.

B. Cooperative juggling

The system can also be run with two quadrocopters playing with one another, set up such that each quadrocopter starts at its opponent's target. In all other respects the scenario is the same as when a human is throwing the ball for a quadrocopter to return.

In Fig. 7 a rally between two quadrocopters is shown, where the ball is kept in the air for 17 consecutive hits. This was after sufficient time had passed for the estimates to settle. The figure shows that the quadrocopters manage to sustain the desired maximum ball height, and keep the impact point approximately at the desired $x \pm 1.5\,\mathrm{m}$, $y = 0$.

The rally was part of a "game" between the two vehicles which lasted for 8 minutes with 160 successful hits over

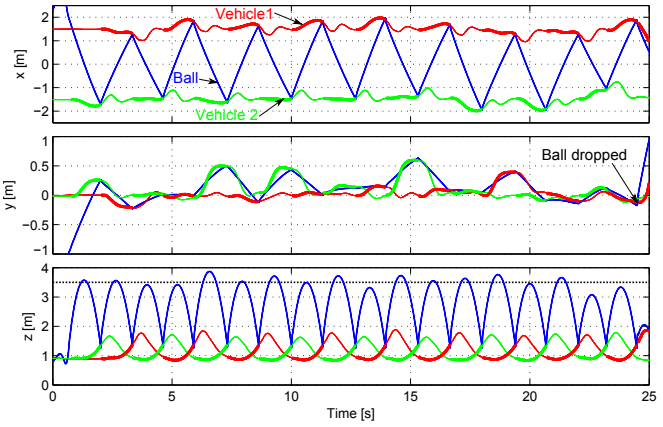

Fig. 7. Position history for two quadrocopters hitting a ball back-and-forth over a $3\,\mathrm{m}$ separation. The blue line is the ball's trajectory, while red and green are for the vehicles. The bold sections are where the quadrocopter is following an interception trajectory, rather than simply returning to a waiting point. The dashed line in $\mathbf{z}$ is the user-defined desired maximum ball height.

24 rallies. The distribution of the rally lengths is shown in Fig. 9. On the histogram, it is interesting to note that the system appears to have a bi-modal distribution, where the ball is either dropped after few hits, or the system manages to sustain the rally for a longer time.

The longest rally ever achieved on the system lasted almost 140 hits, but was not recorded.

C. Solo juggling

During single quadrocopter juggling, the vehicle has the least amount of time between consecutive impacts. Furthermore, at the start of each trajectory (directly after the previous impact), the quadrocopter typically has a large positive vertical speed, and possibly large angles, lateral velocity and angular rates. This makes solo juggling the most challenging task – refer to Section VII for more detail. One juggling rally of seven consecutive hits is shown in

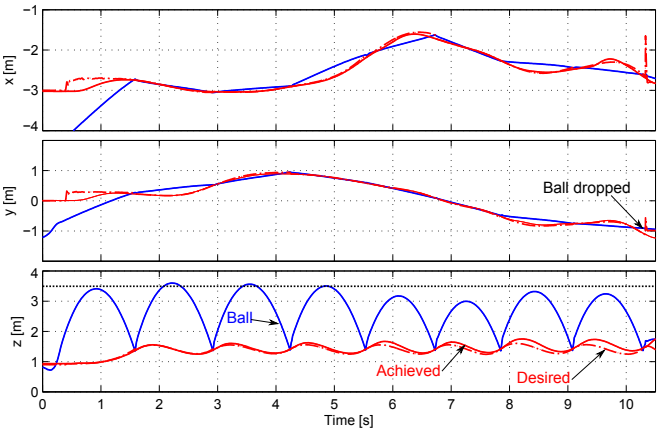

Fig. 8. A single quadrocopter juggling a ball 7 times. The blue line is the ball trajectory, while the solid red line is the vehicle's actual position, and the broken red line the desired position. The dashed line in $\mathbf{z}$ is the desired maximum ball height. Note how the system struggles to maintain this height, see Fig. 10 and Section VII for details.

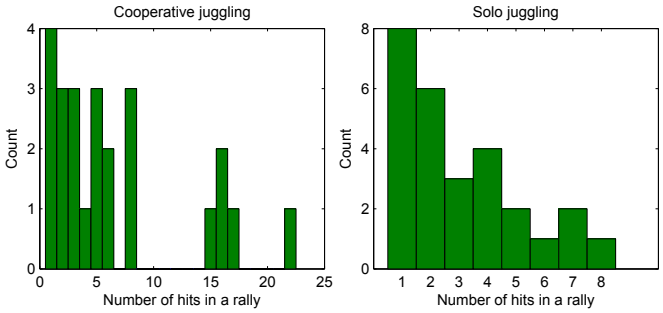

Fig. 9. Histograms showing the system performance when two vehicles hit a ball back-and-forth (left) and when a single vehicle juggles on its own (right). Interesting to note is the bi-modal nature of the distribution when two vehicles play together – where the ball is either dropped early on, or the system manages to sustain a longer rally.

Fig. 8, with a detail of the vehicle's $\mathbf{z}$ trajectory in Fig. 10, also showing the motor commands. On Fig. 8 we notice that the system cannot maintain the desired impact point at $x = y = 0$.

In Fig. 9 a histogram of a single vehicle's juggling performance is shown. The data shows 27 juggling "rallies", lasting an average of 3 hits.

The longest juggling rally ever achieved by the system lasted 14 hits, but was again not recorded.

VII. Failure analysis

We identify three main causes for the system failing to hit the ball: input (motor) saturation, unpredictable bouncing and tracking errors.

A. Input saturation

The trajectory generator does not take input saturation into account, and the generated trajectory might be infeasible. For example, the commanded initial downwards acceleration is often in excess of g, which is unachievable on this system, since each propeller can only produce positive thrust and some thrust is needed to maintain vehicle attitude.

In Fig. 10 we can see the effect of motor saturation. Taking the thrust produced by a propeller as proportional to its rotational speed squared, we expect a speed command which looks like a parabola turned on its side. During the latter stages of the trajectory, we can see this shape, but it is initially saturated. Some motor commands are at the minimum, and the remainder show much larger commands than expected. These commands can be understood as the feedback controller regulating the vehicle's attitude. Since the vehicle's motors can only produce upwards thrust, this has the undesired consequence of producing a net upward force and reducing the vehicle's potential for downwards acceleration.

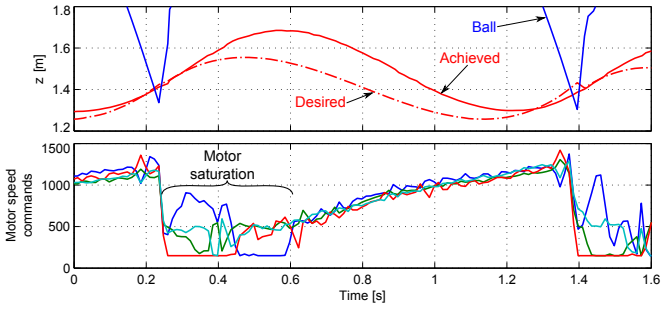

Fig. 10. Motor saturation during solo juggling. On top is shown the system performance when following a commanded trajectory (shown for height only), and below the corresponding motor speed commands, each motor a different colour. The commands are internal, but are linearly related to motor speed. Clearly visible is that after generating the new trajectory, the motor commands are too large, leading to insufficient downwards acceleration. For an affine acceleration, we expect monotonously increasing motor commands.

B. Unpredictable impacts

Similar to how we measure the racket's coefficient of restitution in Section III, we can measure the orientation of the racket's normal for each impact. We notice that this normal deviates from the expected vehicle $\mathbf{z}$-axis – this deviation will cause the ball's bounce to be in a slightly different direction from the expected, in turn leading to aiming errors. A histogram of such deviation measurements is shown in Fig. 11, showing measurements made during the solo juggling experiment of Section VI.

To quantify the effects of such a deflection, we analyse a single quadrocopter during solo juggling, maintaining a maximum ball height h_{max} above the impact point. For simplicity, we assume zero drag. From the conservation of mechanical energy, such a ball will return to the impact height at $\dot{\mathbf{s}}_b^- = \left(0,\, 0,\, -\sqrt{2gh_{max}}\right)$. To return the ball to h_{max}, we want $\dot{\mathbf{s}}_b^+ = -\dot{\mathbf{s}}_b^-$, and noting that the nominal racket normal is $\mathbf{n} = (0,\, 0,\, 1)$, we can calculate the required racket speed using (8) as:

$$\dot{\mathbf{s}}_r = \frac{\beta - 1}{\beta + 1}\dot{\mathbf{s}}_b^- . \tag{34}$$

If the true racket normal ($\mathbf{n}_{defl}$) now deviates from the expected by an angle γ (for convenience taken as a rotation about the y axis), we can calculate the actual post-impact ball velocity ($\dot{\mathbf{s}}_{b,defl}^+$), again using (8):

$$\mathbf{n}_{defl} = (\sin\gamma,\, 0,\, \cos\gamma) \tag{35}$$

$$\dot{\mathbf{s}}_{b,defl}^+ = \sqrt{2gh_{max}} \cdot \left(\sin 2\gamma,\, 0,\, 2\cos^2\gamma - 1\right) \tag{36}$$

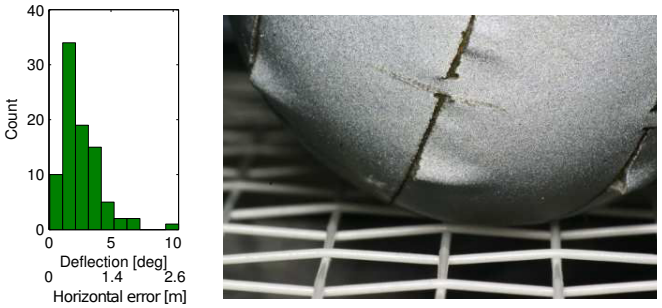

Fig. 11. A histogram showing the angle between the measured normal, and that expected, for a quadrocopter juggling a ball on its own. On the right is a detail showing the uneven surface of the racket, and the shape of the ball. The deflection was measured, and the shown horizontal deflection derived therefrom – refer to the text for details.

We can now calculate the horizontal distance the ball moves before returning to the impact height (Δx), yielding:

$$\Delta x = 4h_{max} \sin 2\gamma \left(2\cos^2 \gamma - 1\right) \tag{37}$$

This equation was used to generate the lower abscissa label in Fig. 11, for a vehicle juggling a ball to a maximum height $h_{max} = 2\,\mathrm{m}$. We can see a 5° deviation leads to a horizontal error of $1.4\,\mathrm{m}$, which would likely be too large for the vehicle to successfully intercept again.

C. Tracking errors

Due to measurement errors and external disturbances, we expect that the vehicle will not achieve a commanded state perfectly. Generally, this is difficult to quantify, but in Fig. 12 the position errors are shown for a vehicle hovering. Here we can see that the vehicle's position error has a mean of $28\,\mathrm{mm}$ – from experience we know that we need to hit a ball within about $50\,\mathrm{mm}$ of the racket centre.

The tracking errors during aggressive flight are much more difficult to analyse, but can be expected to be at least as large as the errors during hover.

VIII. Conclusion

In this paper we have presented a system for a quadrocopter to hit a ball towards a target. This was done by analysing simple models of the ball flight, racket/ball interaction and quadrocopter flight. A Kalman filter was implemented to estimate the ball state, which is needed to predict the impact conditions. Using the impact conditions, the desired quadrocopter state at impact can be calculated, which we combine with affine inputs to move the quadrocopter from an arbitrary initial state to the desired state at impact, under the assumption that the angles remain small.

Strategies were implemented which allow the system to estimate the ball's drag coefficient and the racket's coefficient of restitution, and learn an aiming bias. The combination has been shown to improve the system's performance hitting a ball at a target.

The algorithm was implemented for three different experiments: a single vehicle returning a ball thrown by a human, two vehicles hitting a ball back-and-forth, and a single vehicle attempting to juggle on its own. The performance of the system in each case has been shown, with the first being used to demonstrate the effects of the parameter identification. The juggling performance for two vehicles cooperating was shown to be much better than that of one vehicle on its own, mostly because the vehicles have more time to respond, and start each trajectory in a more favourable position.

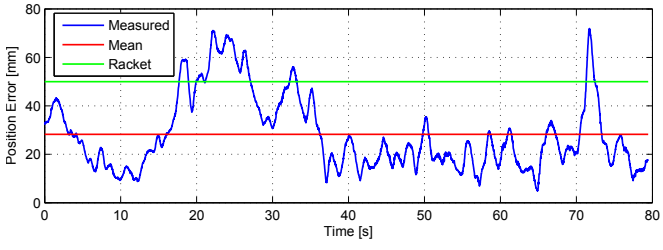

Fig. 12. Total position errors while commanding the vehicle to hover at a fixed location. The blue line shows the distance that the quadrocopter is from the commanded point and the red shows the mean error over this time. The racket has a usable radius of approximately $50\,\mathrm{mm}$, shown by the green line.

The system offers various possibilities for improvement. One can imagine the hitting action encoded as a motion primitive, described by a simple set of parameters which the system can learn to improve so that the resulting motion is closer to the desired (similar to the flips of [16]). Furthermore, a game can be created, like the robot ping-pong of [1], where different strategies (or even completely different vehicles) can be compared in a competitive environment.

References

[1] R. L. Andersson, *A Robot Ping-Pong Player*. The MIT Press, Cambridge, Massachusetts, 1988.

[2] K. Mulling, J. Kober, and J. Peters, "A biomimetic approach to robot table tennis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[3] H. Nakai, Y. Taniguchi, M. Uenohara, T. Yoshimi, H. Ogawa, F. Ozaki, J. Oaki, H. Sato, Y. Asari, K. Maeda, H. Banba, T. Okada, K. Tatsuno, E. Tanaka, O. Yamaguchi, and M. Tachimori, "A volleyball playing robot," in *IEEE International Conference on Robotics and Automations*, 1998.

[4] G. Bätz, M. Sobotka, D. Wollherr, and M. Buss, "Robot basketball: Ball dribbling a modified juggling task," in *Advances in Robotics Research*, T. Kröger and F. M. Wahl, Eds. Springer Berlin Heidelberg, 2009, pp. 323–334.

[5] H. Kitano, M. Asada, I. Noda, and H. Matsubara, "Robocup: robot world cup," *Robotics Automation Magazine, IEEE*, vol. 5, pp. 30 –36, 1998.

[6] M. Bühler, D. E. Koditschek, and P. J. Kindlmann, "A simple juggling robot: Theory and experimentation," in *Proceedings of the First International Symposium on Experimental Robotics*, 1990.

[7] P. Reist and R. D'Andrea, "Bouncing an unconstrained ball in three dimensions with a blind juggling robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[8] A. Schöllig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[9] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[10] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *Int. Symposium on Experimental Robotics*, 2010.

[11] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," in *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, Nov 2010.

[12] J. Nonomura, A. Nakashima, and Y. Hayakawa, "Analysis of effects of rebounds and aerodynamics for trajectory of table tennis ball," in *Proceedings of SICE Annual Conference*, 2010.

[13] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, 2005.

[14] D. Simon, *Optimal State Estimation*. John Wiley & Sons, 2006.

[15] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, Sept.-2 Oct. 2004, pp. 2451–2456 vol.3.

[16] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1642 –1648.

[17] B. W. McCormick, *Aerodynamics Aeronautics and Flight Mechanics*. John Wiley & Sons, Inc, 1995.