Sensing via Collisions: a Smart Cage for Quadrotors with Applications to Self-Localization

Cheng Liu, Roberto Tron

***Abstract*— Applications of micro unmanned aerial vehicles (UAVs) are gradually expanding into complex urban and natural environments. Despite noticeable progress, flying robots in obstacle-rich environments is still challenging. On-board processing for detecting and avoiding obstacles is possible, but at a significant computational expense, and with significant limitations (e.g., for obstacles with small cross sections, such as wires). A low-cost alternative is to mitigate physical contacts through a cage or other similar protective devices. In this paper, we propose to transform these *passive protective devices* into *functional sensors*: we introduce a suspended rim combined with a central base measuring the relative displacement of the rim; we provide a full mechanical design, and derive solutions to the inverse kinematics for recovering the collision direction in real time. As a proof of concept, we show the benefits of this novel form of sensing by embedding it in a traditional particle filter for self-localization in a known environment; our experiments show that localization is possible with a minimal sacrifice in payload capacity.**

I. Introduction

Development activities in multi-rotor aerial robots (and quadrotors in particular) have accelerated dramatically over the recent decades, with mounting research and commercial interests in sensing, surveillance, search operations in dangerous or complex environments, etc. While there has been tremendous progress in developing basic autonomy functionality (e.g., localization, mapping, path planning, robust control, etc.), practical applications still incur significant risks of collisions in cluttered environments. Strategies to counteract this risk fall into two categories: *1)* obstacle detection and avoidance [1]–[3], and *2)* mechanical resilience [4]–[6].

State-of-the-art obstacles-avoidance systems are based on either vision [7], [8], or distance-measuring devices [9]–[11]. Popular implementations rely on global or local maps built via simultaneous localization and mapping or filtering based on onboard sensors [12]–[16], which however have significant limits in representing fine obstacles such as wires and foliage. The second, complementary approach to deal with collisions is to use mechanically resilient systems that attenuate the effect of external forces on flight stability.

Rigid protective accessories such as individual propeller guards are designed to physically separate fragile vehicle components from the environment rather than mitigating impact forces [17]–[19]. More advanced mechanical concepts [20]–[25] absorb energy or isolate impacts through elastic deformations. In both cases, these additional devices reduce the available payload capacity, without directly contributing to main mission objectives.

Both authors are with the Department of Mechanical Engineering, Boston University, 110 Cummington Mall, MA 02215, United States {cliu713,tron}@bu.edu

R. Tron was partially supported by NSF award DCSD-1728277

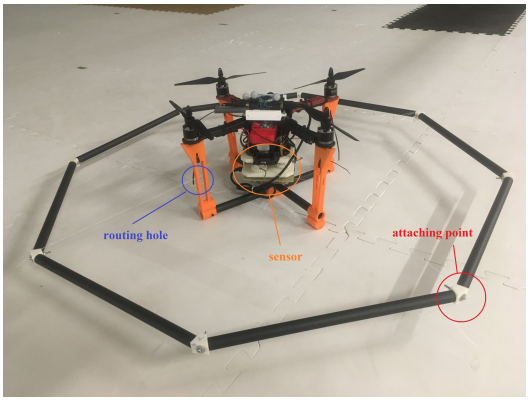


Fig. 1: Prototype of a UAV with the proposed smart cage

Our contributions are aligned with other work [26] that uses additional structures to fundamentally increase the capabilities of quadrotors.

Paper contributions. We introduce a novel mechanical system that combines a protective outer rim with an inner displacement sensor base (Fig. 1). The external rigid protective rim protects the vehicle from external contacts, while its displacement is used to estimate the direction and magnitude of the collision. The contributions of our work are

1) A detailed mechanical and electrical design of the novel system, which translates the motion of the outer rim into the motion of four rotary encoders in the sensor base;
2) The derivation and solution of the inverse kinematics equations necessary to transform the rotary encoder readings into estimates of the collision direction and magnitude. Our solution is non-iterative and can be implemented on a microcontroller (Arduino) in real time.
3) A proof-of-concept validation where we use our sensor for localization in a standard particle filter.

These contribution are illustrated in the following sections. Note that while our experiments show that the sensor alone might be sufficient for localization in a simplified setting, its intended use would be as additional or complementary to other sensors (e.g., vision, inertial, GPS, etc.); e.g., "touch" sensing can be used to integrate standard vision-based solutions in the case of low-light conditions. Our work naturally complements also the work from [27] which uses collision for mapping but visual-inertial odometry for localization.

II. MECHANICAL DESIGN

Our system is divided in a sensor base and an outer rim. The main design challenges are concentrated on the sensor base, which needs to transfer the displacement of the outer rim into the motion of four rotary encoders through cables, springs, and gears. The sensor has four components:

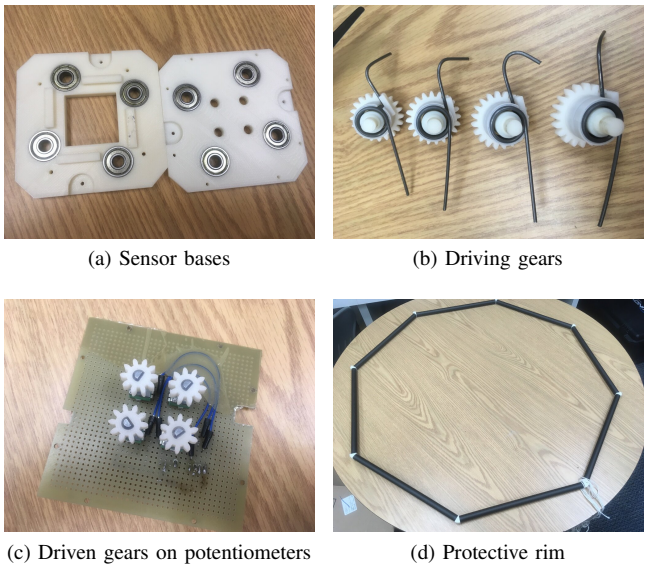
1) An upper and lower base interconnected with struts (Fig.2a): these provide gear housing and sensor support. The two bases have similar layouts, each one with four circular holes designed to house ball bearings for the driving gears. The bottom base includes a large cut-out for reducing weight, and the top base includes four smaller holes through which the shafts of the encoders form the electrical board can attach to the driven gears.

2) Four pairs of gears with torsional springs (Fig. 2b and 2c): The outer rigid rim is attached to the torsional springs through cables, which are routed through holes in the main body of the host vehicle and around coaxial extensions of the gears. The torsional springs are embedded in the driving gears. A gear ratio of 2.0 is used to mechanically amplify the small displacements of the rim (and the cables) during collision, while the springs move the sensor back to rest after collisions.

3) A circuit board with rotary encoders (potentiometers), which interface with the gears. The circuit schematic is shown in Fig. 4. Components `R1` through `R2` refer to the rotary encoders, which are used as voltage dividers. A first connector `JP1` is used for the power supply, while a second connector `JP2` connects the floating contacts of the encoders to the analog inputs on the Arduino.

4) An Arduino board that measures the displacement along the four cables and gear pairs as voltage changes in the analog inputs interfaces, and run the equation solver described in Sec. III to recover the actual 2-D relative rotation and translation of the base with respect to the rim. The power supply and wireless communication for our sensor is derived from the host quadrotor.

The four components stacked in the assembled sensor base are shown in Fig. 3. We take advantage of 3-D printing for fabrication of the two bases and gears, using standard miscellaneous hardware for the other parts. In addition, we designed an outer protective rim (Fig. 2d), which is composed of segments of carbon fiber tubing with 3-D printed joints. Our design makes the rim lightweight but fairly rigid; this is done on purpose, since our goal is to transmit (and dissipate) the energy of the impacts to the springs in the main sensor, not to deformations of the outer rim. The complete sensor is assembled by stacking the components (Fig. 3) and mounting them on the host (Fig. 1), and then running cables (made of fishing line in our prototype) from the outer rim, through the mounting holes, to the springs in the driving gears. The cable is connected to the springs via hooks, and to the outer rim by pressing it between two washers with a screw (this method allows us to manually fix the tension of the cables with just a screwdriver, while avoiding slippage even during collisions). The contact between the fishing line and the 3-D



(a) Sensor bases (b) Driving gears

(c) Driven gears on potentiometers (d) Protective rim

Fig. 2: Components of the system before assembly. Note that, for illustration purposes, the driven gears in (c) are shown mounted directly above the circuit board without the upper base in the middle.

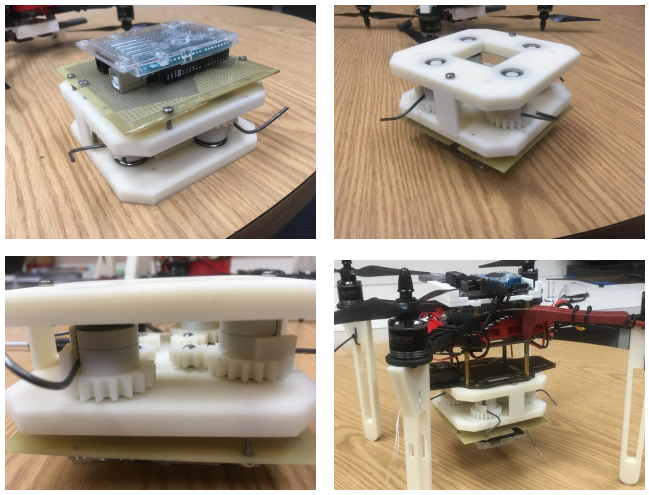


Fig. 3: Different views of the assembled sensor, alone and mounted on the host vehicle.

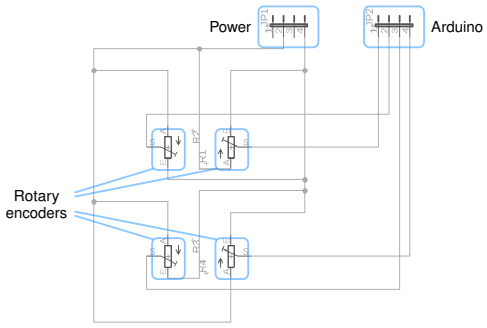


Fig. 4: Electrical circuit schematic

printed routing holes is not a significant source of friction. For proper operation, the cables need to be fairly taut, so that the outer rim is almost level with the routing holes (as a result, the springs will be in tension even at rest).

Remark 1: Although the current design represents a useful proof of concept (see results of the experiments in Sec. V), the system could be improved. For instance, in our prototype, the potentiometer readings present some hysteresis (mainly due to the use of a 3-D printed gear assembly and the springs) that we filter out using simple signal processing.

III. MATHEMATICAL MODEL FOR DISPLACEMENT ANALYSIS

As already mentioned, the Arduino reads changes in voltage in response to displacements of the outer rim. In this section, we derive the 2-D kinematic equations relating the voltages to the relative rotation and translation between the rim and the sensor, and an algorithm for solving them.

We first define two 2-D reference frames: $\mathcal{B}$ (body), rigidly attached to the sensor and the host vehicle, and $\mathcal{C}$ (cage), attached to the outer rim. The two reference frames are defined so that they coincide when the system is at rest, i.e., when there is no collision and no relative displacement between rim and sensor. For modeling purposes, we assume planar motion between the rim and sensor, although, in practice, the rim can show noticeable out-of-plane inclination during collisions and dynamic maneuvers. Despite this, our model can still produce readings that are useful for localization, as demonstrated in the experiments (Sec. V).

Our goal is to use the readings from the potentiometers to solve for the three degrees of freedom of the relative pose between the frames $\mathcal{B}$ and $\mathcal{C}$: two for translation, Δx, Δy, and one for rotation, θ. In practice, we could ignore the rotational component, since, in most practical situations, $\theta \approx 0$. However, we decided to adopt a more general approach for three reasons: *1)* by construction, our equations for obtaining the translation are still valid when θ is known, *2)* if a more precise sensor becomes available in the future, our solver would still be valid, and *3)* discuss a fundamental symmetry ambiguity (between θ and $-\theta$) that does not appear when $\theta = 0$.

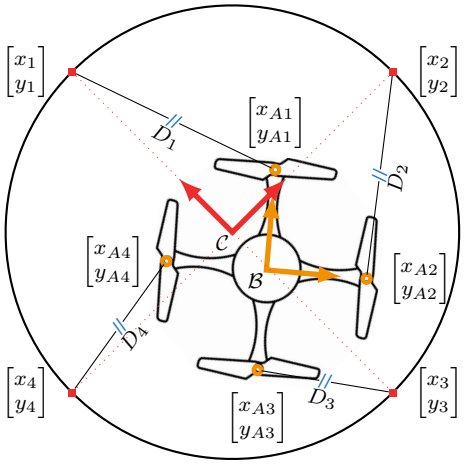


Fig. 5: Illustration of the reference frames for the cage $\mathcal{C}$ (outer rim) and the body $\mathcal{B}$ (host vehicle), connected via cables –//– between routing holes ○ and attaching points ▪.

Given the notation above, the coordinate transformation from the rim frame $\mathcal{C}$ to the vehicle frame $\mathcal{B}$ is given by the following rigid body transformation:

$$\begin{bmatrix} x^{\mathcal{B}} \\ y^{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x^{\mathcal{C}} \\ y^{\mathcal{C}} \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{1}$$

where x and y are the coordinates of a given point.

We denote the constant distance between the origin of $\mathcal{B}$, and each routing hole by $L_{\mathcal{B}}$, and the constant distance between the origin of $\mathcal{C}$, and the attachment point of the cable as $L_{\mathcal{C}}$, see Fig. 5 for a schematic illustration. We then have that the coordinates of the four routing holes in the base frame $\mathcal{B}$ are:

$$\begin{aligned} \begin{bmatrix} x_1^{\mathcal{B}} \\ y_1^{\mathcal{B}} \end{bmatrix}^{\mathrm{T}} &= \begin{bmatrix} 0 \\ L_{\mathcal{B}} \end{bmatrix}^{\mathrm{T}}, & \begin{bmatrix} x_2^{\mathcal{B}} \\ y_2^{\mathcal{B}} \end{bmatrix}^{\mathrm{T}} &= \begin{bmatrix} L_{\mathcal{B}} \\ 0 \end{bmatrix}^{\mathrm{T}}, \\ \begin{bmatrix} x_3^{\mathcal{B}} \\ y_3^{\mathcal{B}} \end{bmatrix}^{\mathrm{T}} &= \begin{bmatrix} 0 \\ -L_{\mathcal{B}} \end{bmatrix}^{\mathrm{T}}, & \begin{bmatrix} x_4^{\mathcal{B}} \\ y_4^{\mathcal{B}} \end{bmatrix}^{\mathrm{T}} &= \begin{bmatrix} -L_{\mathcal{B}} \\ 0 \end{bmatrix}^{\mathrm{T}}, \end{aligned} \tag{2}$$

with similar expressions for the coordinates of the four attachment points of the cable in the cage frame $\mathcal{C}$:

$$\begin{aligned} \begin{bmatrix} x_{A1}^{\mathcal{C}} \\ y_{A1}^{\mathcal{C}} \end{bmatrix} &= \begin{bmatrix} 0 \\ L_{\mathcal{C}} \end{bmatrix}^{\mathrm{T}}, & \begin{bmatrix} x_{A2}^{\mathcal{C}} \\ y_{A2}^{\mathcal{C}} \end{bmatrix} &= \begin{bmatrix} L_{\mathcal{C}} \\ 0 \end{bmatrix}^{\mathrm{T}}, \\ \begin{bmatrix} x_{A3}^{\mathcal{C}} \\ y_{A3}^{\mathcal{C}} \end{bmatrix} &= \begin{bmatrix} 0 \\ -L_{\mathcal{C}} \end{bmatrix}^{\mathrm{T}}, & \begin{bmatrix} x_{A4}^{\mathcal{C}} \\ y_{A4}^{\mathcal{C}} \end{bmatrix} &= \begin{bmatrix} -L_{\mathcal{C}} \\ 0 \end{bmatrix}^{\mathrm{T}}. \end{aligned} \tag{3}$$

Under collision, the length of the cable between the sensor and the rim effectively changes (since it moves together with the spring), while the cables also bend at the routing holes, causing the outer rim to rotate and translate. We therefore define four distances D_i, $i \in \{1, \ldots, 4\}$, which represent the distances between the points in $\mathcal{B}$ defined in (2) and the corresponding points in $\mathcal{C}$ defined in (3). More precisely, we can transform the points (3) to the $\mathcal{B}$ frame using (1), and then computing the squared norm of the differences, obtaining:

$$D_1^2 = (L_{\mathcal{C}} \sin\theta - \Delta x)^2 + (L_{\mathcal{B}} - L_{\mathcal{C}} \cos\theta - \Delta y)^2, \tag{4a}$$
$$D_2^2 = (L_{\mathcal{B}} - L_{\mathcal{C}} \cos\theta - \Delta x)^2 + (L_{\mathcal{C}} \sin\theta + \Delta y)^2, \tag{4b}$$
$$D_3^2 = (L_{\mathcal{C}} \sin\theta + \Delta x)^2 + (L_{\mathcal{B}} - L_{\mathcal{C}} \cos\theta + \Delta y)^2, \tag{4c}$$
$$D_4^2 = (L_{\mathcal{B}} - L_{\mathcal{C}} \cos\theta + \Delta x)^2 + (L_{\mathcal{C}} \sin\theta + \Delta y)^2. \tag{4d}$$

We assume that the distances $\{D_i\}$ can be determined by the onboard sensor using the following lumped parameter model:

$$D_i = L_{\mathcal{C}} - L_{\mathcal{B}} + \Delta D_i, \qquad \Delta D_i = \alpha v_i + \beta, \tag{5}$$

for $i \in \{1, \ldots, 4\}$, and where the $\{\Delta D_i\}$ express the change in length with respect to the length at rest (given by $L_C - L_B$), v_i are the analog readings from the Arduino, which depend on the encoders' rotations, and α, β subsume the effects of the initial readings, and angular range of the potentiomenters, the gear ratio, and the mechanical arm of where each cable

attaches to the corresponding spring and gear. The parameters α and β are manually adjusted in a separate calibration.

Given the distances D_i (obtained from the Arduino inputs via (5)), our goal is to solve the over-determined system of quadratic equations (4) to find Δx, Δy and θ. To proceed, we first notice by inspection that the linear combination (4a) − (4b) + (4c) − (4d) gives a constant equation (i.e., all the variable are eliminated), and represents an invariant of the system. We therefore extract the following three linearly independent equations: the sum of (4a) through (4b), and the differences (4c) − (4a), (4d) − (4b), resulting in

$$\sum_{i=1}^{4} D_i^2 - L_{\mathcal{B}}^2 + L_{\mathcal{C}}^2 - 2L_{\mathcal{B}}L_{\mathcal{C}}\cos\theta + \Delta x^2 + \Delta y^2 = 0, \quad \text{(6a)}$$

$$D_3^2 - D_1^2 - 4\big((L_{\mathcal{B}} - L_{\mathcal{C}}\cos\theta)\Delta y + L_{\mathcal{C}}\Delta x \sin\theta\big) = 0, \quad \text{(6b)}$$

$$D_4^2 - D_2^2 - 4\big((L_{\mathcal{B}} + L_{\mathcal{C}}\cos\theta)\Delta x + L_{\mathcal{C}}\Delta y \sin\theta\big) = 0, \quad \text{(6c)}$$

$$\cos^2\theta + \sin^2\theta = 1, \quad \text{(6d)}$$

where we added the last equation from a known trigonometric identity. Considering $\cos\theta$ and $\sin\theta$ as two independent variables constrained by (6d), equation (6) is a system of polynomial (in fact, quadratic) equations in four variables. Applying the method of [28], we can solve this new system of equations with the following steps:

1) Use a Gröbner basis to algebraically eliminate all the variables except $\cos\theta$.
2) Express possible solutions for $\cos\theta$ as the eigenvalues of the associated *action matrix*, whose characteristic equation gives a quartic polynomial in $\cos\theta$.
3) If the equation is satisfied by the special case $\cos\theta = 1$ ($\theta = 0$), skip to step *5)*.
4) Solve the quartic polynomial using Ferrari's method [29]. Given the symmetries in the original system (6) and in the action matrix, this quartic polynomial will have one real solution (with double multiplicity) determining $\cos\theta$, and thus two opposite values for θ.
5) Substitute θ in (6b), (6c), and solve for Δx, Δy (notice that this is a linear system once θ is known).

In general, we will obtain two solutions: this is expected, since two mirrored configurations with θ and $-\theta$, and, respectively, Δx and $-\Delta x$ will produce the same distances $\{D_i\}$; this ambiguity could be resolved using complementary inertial measurements; in our experiments, we have $\theta \approx 0$, so we do not consider this ambiguity. The important point about the steps *1)-5)* above is that they are non-iterative, and can be mostly carried out with simple arithmetic operations; hence, they can be easily implemented directly on the Arduino board, and run in real time. Nonetheless, note that we used [28] only for step *1)* above, with custom implementations for the other steps, in order to avoid numerical routines that cannot be efficiently implemented on a microcontroller. The Arduino publishes the results in the Robotic Operating System (ROS) using the `rosserial` package via the host vehicle.

IV. LOCALIZATION USING PARTICLE FILTERING

In this section, we follow a typical particle filtering formulation [30], summarized in Fig. 6), focusing mostly on the elements that are specialized to our setting; we assume a simplified setting where the rotation between the vehicle's and the world's reference frames are known (e.g., using measurements from an onboard Inertial Measurement Unit and magnetometer). Our goal is not to propose a new particle filter, but simply show how our sensor could be incorporated.

First, we assume a first-order integrator dynamics as,

$$\vec{x}_k^{\text{sys}} = \vec{x}_{k-1}^{\text{sys}} + \vec{v}_{k-1} + \vec{u}_{k-1}, \quad (7)$$

where $\vec{u}$ refers to the motion command given to the vehicle, and $\vec{v}$ is Gaussian noise; k denotes the discrete time.

We use collisions as our observations for the filter update. Our observation model is based on two quantities derived by the displacement components Δx, Δy (as already mentioned, we make the practical approximation $\theta = 0$): the influence distance D_{infl}, and the surface normal vector $\vec{N}_{\text{infl}}$ at the point of collision between the obstacle and the vehicle:

$$D_{\text{infl}} = D_0 - \sqrt{\Delta x^2 + \Delta y^2}, \quad (8)$$

$$\vec{N}_{\text{infl}} = \frac{\begin{bmatrix}\cos\theta_{\text{vehicle}} & -\sin\theta_{\text{vehicle}} \\ \sin\theta_{\text{vehicle}} & \cos\theta_{\text{vehicle}}\end{bmatrix}\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}}{\sqrt{\Delta x^2 + \Delta y^2}}, \quad (9)$$

where D_0 is the rim radius, and $\theta_{vechile}$ is the angle between the vehicle's and world's reference frames.

We then define a weight function φ to model the relation between the measured distance of influence D_{infl}, and the distance between a particle and the nearest obstacle, D_{obs}:

$$\varphi(D_{\text{obs}}, D_{\text{infl}}) = \begin{cases} \max\left[0, 1 - \left|\dfrac{D_{\text{obs}}}{D_{\text{infl}}} - 1\right|\right] & \text{if collision,} \\ \max\left[0, \min\left(1, \dfrac{D_{\text{obs}}}{D_{\text{infl}}}\right)\right] & \text{otherwise.} \end{cases} \quad (10)$$

The two cases are pictorially shown in Fig. 7. Intuitively, if the sensor measures that the outer rim is not in collision (Fig. 7a), the particles that are close to obstacles will be less

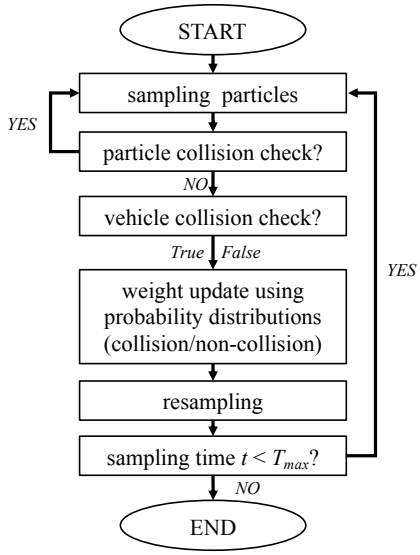
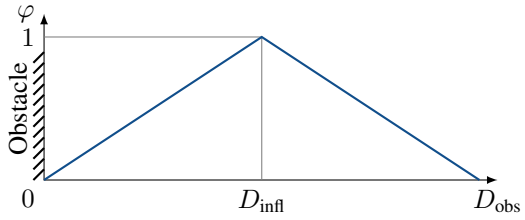
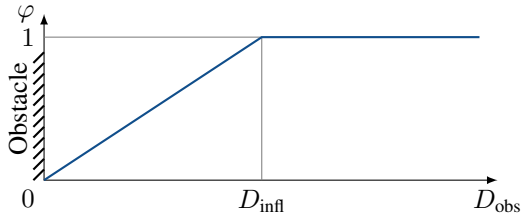


Fig. 6: Particle filtering workflow



(a) $\varphi(D_{\text{obs}}, D_{\text{infl}})$, no collision



(b) $\varphi(D_{\text{obs}}, D_{\text{infl}})$, collision

Fig. 7: Probability distributions for particle weight given collision observation

likely; conversely, if the sensor measures a collision (Fig. 7b), particles that have a distance D_{obs} closer to the expected D_{infl} will be more likely, with zero probability at $D_{\text{obs}} = 0$ and $D_{\text{obs}} \geq 2D_{\text{infl}}$.

We also introduce an additional weight function ψ modeling the relation between the measured normal of influence N_{infl}, and the corresponding quantity for a particle, N_{obs}:

$$\psi(\vec{N}_{\text{obs}}, \vec{N}_{\text{infl}}) = \begin{cases} \frac{\vec{N}_{\text{obs}}^{\text{T}} \vec{N}_{\text{infl}} + 1}{2} & \text{if both in collision,} \\ 1 & \text{otherwise.} \end{cases} \quad (11)$$

Intuitively, if the sensor reveals a collision, we give more importance to particles that also imply a collision with a normal vector similar to the direction measured by the sensor; conversely, if there is no collision (either signaled by the sensor, or implied by the particle) we do not have normals to compute, so we set the weight to the maximum value. The final weight for the particle is given by multiplying (10) with (11), and is used for resampling as described in Fig. 6.

V. EXPERIMENTAL VALIDATION

A. Experimental Set Up

Our experimental setup is shown in Fig. 9. We created vertical obstacles in Plexiglass with $80/20$ aluminum supports, and the tests have been carried out in the motion capture (mocap) volume in the BU Robotics Lab. We use the mocap system to ground-truth geometric information the obstacles (vertices, normal vectors) and the actual UAV trajectory.

As the host, we use a custom quadrotor platform, using a Pixhawk flight controller for low-level stabilization, and an Odroid XU 4 running Ubuntu 16.04 LTS and ROS Kinetic for high-level processing. The Odroid board communicates with the Pixhawk and the Arduino through serial interfaces, and with a base station using WiFi networking. We use a custom message, published at a constant rate, containing the most recent sensor measurement from the Arduino and the most recent pose from the mocap system, for synchronization purposes. This topic is recorded as a `rosbag` file.

The particle filtering algorithm of Sec. IV is implemented in Matlab, and is run on the data from the `rosbag` file. The orientation of the vehicle is assumed known (from the mocap data), and the filtering algorithm estimates the position alone, which is then compared against the mocap ground truth (again, this is a proof of concept for verifying the sensor design, not for achieving state-of-the-art localization).

Fig 9 shows snapshots of the flight path, and Fig. 10 shows the map of the environment with the evolution of the particles (starting from a uniform distribution) together with the ground truth position. The flight path follows waypoints that lead the vehicle to two collisions with two separate obstacle surfaces.

B. Results

We compare the average position of the particles, $\vec{x}_k$, with the ground truth translation, $\vec{x}_k^{\text{sys}}$, and for each time step k we compute the 2-D Euclidean distance error $e_k = \|\vec{x}_k - \vec{x}_k^{\text{sys}}\|$, shown in Fig. 8. At the beginning, the position estimate is near the center of the map due to the uniform initialization, leading to high error. The error decreases rapidly after the first collision (around $k = 120$). Afterward, the error starts to climb again, up to due to the dead-reckoning performed by the filter until the next collision (around $k = 1380$). The very low absolute values for the error after each collision are somewhat due also the fact that the obstacle surfaces have dimensions comparable to the one of the outer rim of our sensor; larger obstacles would leave more uncertainty longitudinally to the edge. Fig. 10 shows the evolution of the filter in more detail. Starting from a uniform distribution (Fig. 10l), as the vehicle gets closer to the obstacle, the particles around the obstacle are gradually eliminated (Fig. 10b), because each particle weight is updated using the no-collision cases in (10) and (11), assigning low weight to particles whose distance is less than the rim's radius (i.e., that imply a collision), which are then effectively rejected by the resampling. After starting to sense the first impact with the obstacle (Figs.9b, 10c, and 10d) the particles rapidly coalescence; in particular, notice that since our sensor can estimate the direction of the impact normal, only the particles near obstacle surface with the correct orientation are kept. As the vehicle moves away

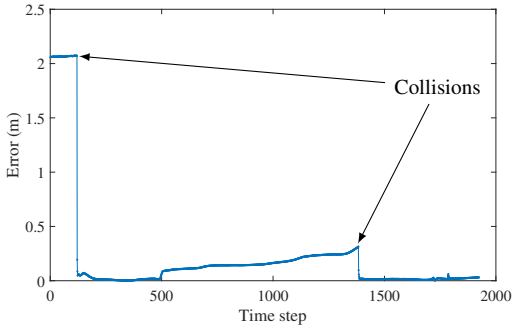
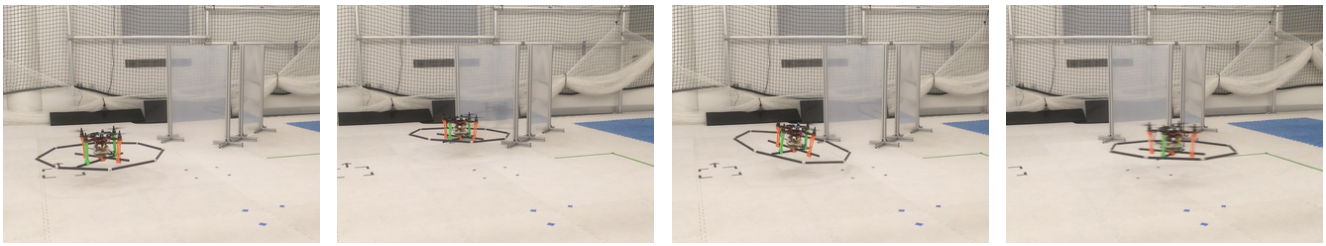


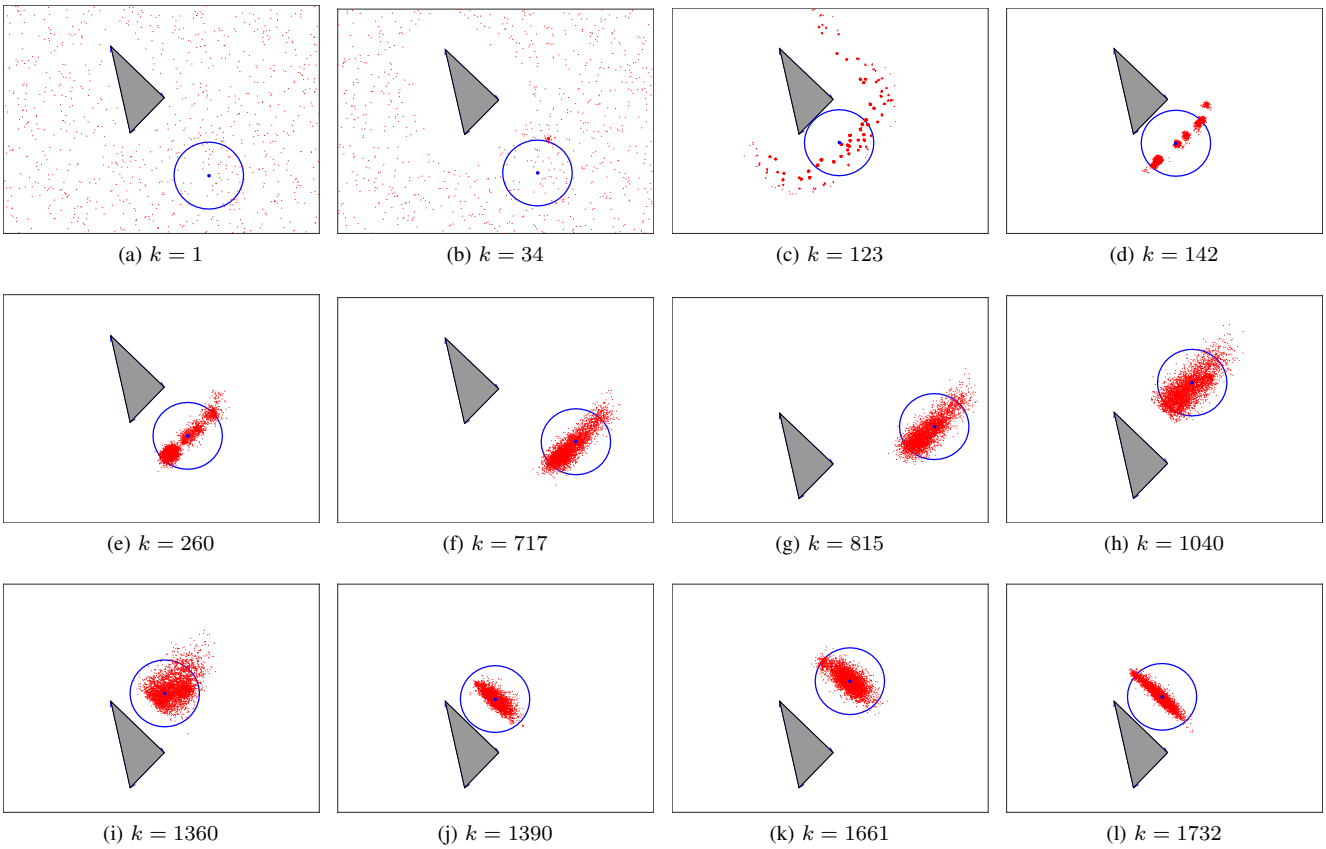
Fig. 8: Vehicle position error (estimated versus ground truth)



(a) Waypoint 1 (b) Waypoint 2 (c) Waypoint 3 (d) Waypoint 4

(e) Waypoint 5 (f) Waypoint 6 (g) Waypoint 7 (h) Waypoint 8

Fig. 9: Experimental setup (plexiglass walls fixed on the ground using aluminium rails) and flight path of the host UAV



(a) $k = 1$ (b) $k = 34$ (c) $k = 123$ (d) $k = 142$

(e) $k = 260$ (f) $k = 717$ (g) $k = 815$ (h) $k = 1040$

(i) $k = 1360$ (j) $k = 1390$ (k) $k = 1661$ (l) $k = 1732$

Fig. 10: Map of the environment (gray triangle), position of the vehicle and outer rim (blue point and circle), and evolution of the particle swarm (red dots). The particles are initialized using a uniform distribution.

(Figs. 10e-10g), we return to a no-collision situation (Figs. 9c- to 9e), which leads the cluster of particles to disperse (i.e., as one would expect, the variance of the estimate increases). During the second collision (Figs. 9f-9h), the particles once again become denser. Note also that, as expected the variance captured by the particle density is reduced in the direction normal to the obstacle during both collisions.

In summary, the experiment show that a particle filter can use the impact magnitude and direction of collision measured with the proposed sensor to perform localization, as long as a sufficient number of collisions is maintained.

VI. Conclusion

We propose a collision protection system for multirotor vehicles that works also as a form of touch sensor. We described the mechanical, electrical, and algorithmic components of our system. In particular, we derive a method to estimate the relative translational and rotational displacement of the outer rim with respect to the sensor base. We include a flight experiment to show the feasibility of using our sensor for localization via particle filtering. Going forward, the output of the sensor (e.g., D_{infl}, N_{infl}) could be used not only for localization, but also for more precise control. The precision of the sensor could be improved by using load cells instead of gears, springs, and potentiometers (in the current prototype the sensitivity of the sensor decreases in the directions that are not aligned with the axes). Finally, and we believe that our approach could be extended to a 3-D cage for full 3-D sensing.

References

[1] X Wang, V Yadav, and S.N. Balakrishnan. Cooperative uav formation flying with obstacle/collision avoidance. *IEEE Transactions on Control Systems Technology*, 15(4):672–679, 2007.

[2] S. Temizer, M. T. Kochenderfer, L. P. Kaelbling, T. Lozano-Perez, and J. K. Kuchar. Collision avoidance for unmanned aircraft using markov decision processes. *AIAA Guidance, Navigation, and Control Conference*, pages 1–21, 2010.

[3] Luis Mejias, S. McNamara, J. Lai, and Jason Ford. Vision-based detection and tracking of aerial targets for uav collision avoidance. *Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 87 – 92, 11 2010.

[4] Adrien Briod, Adam Klaptocz, Jean-Christophe Zufferey, and Dario Floreano. The airburr: A flying robot that can exploit collisions. *2012 ICME International Conference on Complex Medical Engineering, CME 2012 Proceedings*, pages 569–574, 07 2012.

[5] Mirko Kovac. Learning from nature how to land aerial robots. *Science*, 352(6288):895–896, 2016.

[6] Carl Salaan, Yoshito Okada, Shoma Mizutani, Takuma Ishii, Keishi Koura, Kazunori Ohno, and Satoshi Tadokoro. Close visual bridge inspection using a uav with a passive rotating spherical shell. *Journal of Field Robotics*, 35, 02 2018.

[7] William Green and Paul Oh. Optic-flow-based collision avoidance. *IEEE Robotics and Automation Magazine*, 15:96 – 103, 04 2008.

[8] Paul Merrell, Lee Dah-Jye, and Randal Beard. Obstacle avoidance for unmanned air vehicles using optical flow probability distributions. *Proceedings of SPIE - The International Society for Optical Engineering*, 12 2004.

[9] Young Kwag and Chul Chung. Uav based collision avoidance radar sensor. *IEEE International Geoscience and Remote Sensing Symposium*, pages 639–642, 01 2007.

[10] Sachin Modi, Pravin Chandak, Vidya Murty, and Ernest Hall. Comparison of three obstacle-avoidance methods for a mobile robot. *Proceedings of SPIE - The International Society for Optical Engineering*, 4572, 10 2001.

[11] Nils Gageik, Paul Benz, and Sergio Montenegro. Obstacle detection and collision avoidance for a uav with complementary low-cost sensors. *IEEE*, 3:1–1, 01 2015.

[12] Shaojie Shen, Nathan Michael, and Vijay Kumar. Autonomous multi-floor indoor navigation with a computationally constrained mav. *IEEE International Conference on Robotics and Automation*, pages 20 – 25, 06 2011.

[13] Daniel Magree, John G. Mooney, and Eric N. Johnson. Monocular visual mapping for obstacle avoidance on uavs. *Journal of Intelligent & Robotic Systems*, 74(1):17–26, Apr 2014.

[14] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam-based navigation for autonomous micro helicopters in gps-denied environments. *J. Field Robotics*, 28:854–874, 11 2011.

[15] Markus Achtelik, Simon Lynen, Stephan Weiss, Laurent Kneip, Margarita Chli, and Roland Siegwart. Visual-inertial slam for a small helicopter in large outdoor environments. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2651–2652, 2012.

[16] Girish Chowdhary, Eric Johnson, Daniel Magree, Allen Wu, and Andy Shein. Gps-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft. *Journal of Field Robotics*, 30, 05 2013.

[17] Kenjiro Tadakuma, Riichiro Tadakuma, and Jose Berengeres. Development of holonomic omnidirectional vehicle with "omni-ball": spherical wheels. pages 33 – 39, 12 2007.

[18] Arash Kalantari and Matthew Spenko. Modeling and performance assessment of the hytaq, a hybrid terrestrial/aerial quadrotor. *Robotics, IEEE Transactions on*, 30:1278–1285, 10 2014.

[19] Gabriella Caroti, Andrea Piemonte, Isabel Martínez-Espejo Zaragoza, and Giacomo Brambilla. Indoor photogrammetry using uavs with protective structures: Issues and precision tests. 2018.

[20] Adrien Briod, Przemyslaw Mariusz Kornatowski, Jean-Christophe Zufferey, and Dario Floreano. A collision-resilient flying robot. *J. Field Robotics*, 31:496–509, 2014.

[21] C.D. Onal, Michael Tolley, Robert Wood, and Daniela Rus. Origami-inspired printed robots. *IEEE/ASME Transactions on Mechatronics*, 20:1–8, 12 2014.

[22] Shoma Mizutani, Yoshito Okada, Carl Salaan, Takuma Ishii, Kazunori Ohno, and Satoshi Tadokoro. Proposal and experimental validation of a design strategy for a uav with a passive rotating spherical shell. pages 1271–1278, 09 2015.

[23] Carl Salaan, Kenjiro Tadakuma, Yoshito Okada, Eri Takane, Kazunori Ohno, and Satoshi Tadokoro. Uav with two passive rotating hemispherical shells for physical interaction and power tethering in a complex environment. pages 3305–3312, 05 2017.

[24] Pooya Sareh, Pisak Chermprayong, Marc Emmanuelli, Haris Nadeem, and Mirko Kovac. Rotorigami: A rotary origami protective system for robotic rotorcraft. *Science Robotics*, 3:eaah5228, 09 2018.

[25] Yash Mulgaonkar, Gareth Cross, and Vijay Kumar. Design of small, safe and robust quadrotor swarms. pages 2208–2215, 2015.

[26] Bruno Gabrich, Guanrui Li, and Mark Yim. ModQuad-DoF: A novel yaw actuation for modular quadrotors. pages 8267–8273, 2020.

[27] Yash Mulgaonkar, Wenxin Liu, Dinesh Thakur, Kostas Daniilidis, Camillo J Taylor, and Vijay Kumar. The Tiercel: A novel autonomous micro aerial vehicle that can map the environment by flying into obstacles. pages 7448–7454, 2020.

[28] Zuzana Kukelova, Martin Bujnak, and Tomás Pajdla. Automatic generator of minimal problem solvers. 2008.

[29] Gerolamo Cardano. The rules of algebra: (ars magna). *Dover Publications*, 2007.

[30] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. *The MIT Press*, 2005.

[31] Justin Thomas, Giuseppe Loianno, Koushil Sreenath, and Vijay Kumar. Toward image based visual servoing for aerial grasping and perching. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2113–2118, 05 2014.