Noname manuscript No.
(will be inserted by the editor)

Development of entertainment balloon robot system for an indoor event venue

Hiroya Nagata · Stefan Walke · Soichiro Yokoyama · Tomohisa Yamashita · Hiroyuki Iizuka · Masahito Yamamoto · Keiji Suzuki · Hidenori Kawamura

Received: date / Accepted: date

Abstract This project revolves around the combination of different state of the art concepts to create balloon robots which can be used for entertainment purposes. The goal is having multiple robots perform a choreographed flight on preset paths to make use of the "dead space" over the crowd of an event, either as part of the show or displaying information and advertisements. As the balloon is flying on a defined trajectory, a tracking system for its position is needed. For this purpose, several infrared cameras monitor the position of a marker attached to the balloon. The main challenge is overcoming the instabilities of the system to ensure a smooth and precise flight, resulting from the balloons structure: The balloon is filled with helium to counteract the forces of gravity and therefore minimize the work needed to keep its momentum. Finding a way to achieve this optimization and the precision mentioned beforehand is the task that we will present and solve in this paper.

Keywords Balloon robots · PID controlling · 3D positioning

1 Introduction

In recent years, the technology involved in hosting live events and concerts has steadily progressed. From advancements in quality, spectacular stage setups and interactions to holographic characters on stage. However, there is one thing that remained constant, the crowd area and for this project more relevant the space above the crowd is mostly unused. With this project, we aim to create a possibility to use said space.

Researchers have been studying applications of balloon/blimp robots for quite some time, including entertainment purpose [1]. At the same time, different control methods that can be applied to this area have been researched. These include using PID [2] and/or learning [3] controllers for landing control which could be used for recharging. Furthermore keeping the unit stationary utilizing visual feedback [4] and cooperative control by multiple neural networks [5] are topics of interst for the problem presented in this paper.

The balloon robot we develop (Fig. 1) could be used for subtle display of ads throughout the event or enhancement of the experience by providing ambient lighting and enhancing the atmosphere. In its nature, a smoothly flying balloon is less invasive in the visual perception than for example

Fig. 1 Balloon robot we developed

This work was presented in part at the 22nd International Symposium on Artificial Life and Robotics, Oita, Japan, January 19-21, 2017.

H. Nagata · S. Walke · S. Yokoyama · T. Yamashita · H. Iizuka · M. Yamamoto · H. Kawamura
Graduate School of Information Science and Technology, Hokkaido University, North 14, West 8, Kita-ku, Sapporo 060-8628, Japan
Tel.: +81-11-706-6498
E-mail: nagata@complex.ist.hokudai.ac.jp

K. Suzuki
Future University Hakodate, Hakodate, Hokkaido, Japan

a drone. Of course, this comes at a cost: A helium filled balloon is very susceptible to environmental influences and difficult to control due to its stability properties if well balanced. It therefore requires precise engineering, calculations and tuning to work as intended.

We introduce an experimental setup, that currently supports up to three balloons, but can in theory be expanded to an arbitrary amount of robots, flying simultaneously along predefined paths. These can be either sets of waypoints or mathematical descriptions.

2 Physical setup

The hardware setup for this experiment consists of the balloons, the camera setup for tracking and the main control hub (MCH). We will not go into each of the components and discuss their structure and technical realization.

2.1 Balloon

All three balloons currently in use are structurally identical. The hull is a sphere with 2m in diameter, and made of 0.25mm PVC, which was chosen due to its durability. Earlier prototypes were made of rubber. The choice of material is a surprisingly non-trivial problem. While the current material provides a sturdier structure, and is ideal for testing and frequent inflating and deflating it also adds a lot of weight to the balloon, which leads to the center of mass shifting upwards, towards the center of the balloon, which in turn causes roll and pitch movements to be harder to control, compared to a setup where the center of mass is very low, close to the control unit. The total mass including the filled helium gas is approximately 5.2kg.

The balloon is driven by a total of six rotors units: four controlling the movement in the x-z plane and yaw movements, two for height control. As shown in Fig. 2, they are symmetrically arranged and located at the equator of the balloon.

A rotor unit (Fig. 3) consists of a Turnigy 1000mAh 2S 20C Li-Poly battery, a Hobbyking XC-10A Electric Speed Controller, a Cosmotech CT2211-2200 brushless motor, with a custom four-blade rotor attached and a custom 3D-printed casing. To connect the rotor unit to the control unit we use a standard LAN cable to bundle the signals.

The other part that is attached to the balloon along the same horizontal axis as the rotor units are the antennas (Fig. 4.) Featuring another custom casing, each of the 2 units contains a 5GHz IEEE802.11a Wi-Fi antenna as well as an MU-1-429 radio antenna on one of the units and an MU-1-1216 one on the other. This is done, to ensure a stable connection. As the large area of operation and the wanted precision of control, having only one type of connection would not suffice. The antenna units are connected via standard USB 2.0.

Finally, onto the control unit, shown in Fig. 5. Its core is a Raspberry Pi 2 Model B, running code for the coordinated execution of the flight algorithm driven by the position data it receives from the MCH. It is supported by a PWM signal generator, to generate the appropriate signals for the rotors, since the Raspberry Pi only features one physical PWM output and software PWM signals are very computationally expensive. Furthermore, the control unit has more custom made PCBs to bundle the signals and send them via the mentioned LAN cable. On the bottom of the casing multiple ultra-bright 940nm IR-LEDs are mounted on a half sphere, to ensure visual contact to the cameras at any point in time for tracking, which we will discuss in more detail in sections 2.2 and 3.1. The whole unit is powered by of the mentioned batteries for the Raspberry Pi and one specifically for the IR marker.

Lastly there is a MPU-9250 chip mounted on a custom PCB. This sensor unit features a compass, an accelerometer and a gyroscope with three axes each. These are used to compensate for the suboptimal update speed of the camera tracking we will discuss in the next section. Note, that due to the different voltages needed by all the parts several con-

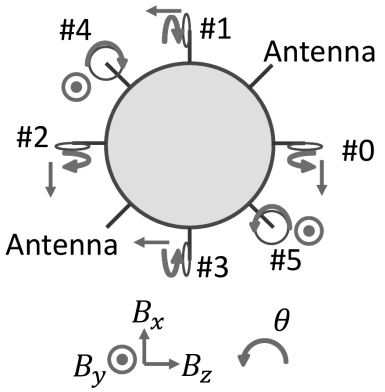

Fig. 2 Placement of rotor units, arrows indicating the positive rotation and thrust directions

Fig. 3 Rotor unit

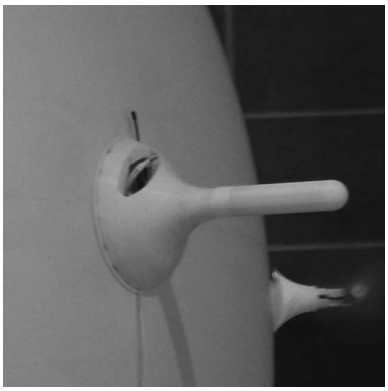

Fig. 4 Antenna unit

Fig. 5 Control unit

verters of different varieties were used, which we will not discuss in detail.

2.2 Camera array and MCH

Moving on to the cameras used for tracking the IR marker mentioned beforehand. There is a total of six Baumer VLG-24M cameras with VS-0814H1 lenses equipped with IR filters (cutoff wavelength 850nm) connected to a Gigabit switch. In theory with good calibration only two cameras can be enough to perform 3D localization, however due to random fluctuations, ambient light and the wanted precision we decided to aim to always maintain line of sight with at least 3-4 cameras. Two more cameras are set up to provide further stability, since positional relationships of the balloons can block the line of sight. Fig. 6 shows an example spatial arrangement of the cameras.

We decided to use infrared light, because the application of the finished product would be indoor event venues with many changings in lighting, so anything in the visible spectrum cannot be used. Furthermore, there is no GPS available indoors and a recreation would not be ergonomic, so there were only few options left and we decided to go with infrared light, since it fits all the criteria we need and is readily available.

The MCH consists of a Dell Alienware15, functioning as the main computing unit for image processing and path

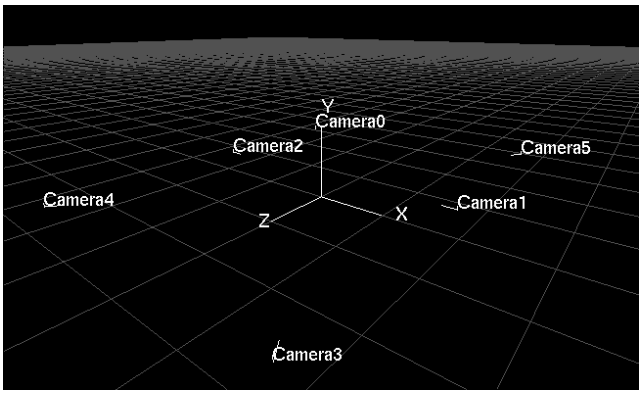

Fig. 6 Spatial arrangement of cameras, grid size is 5m

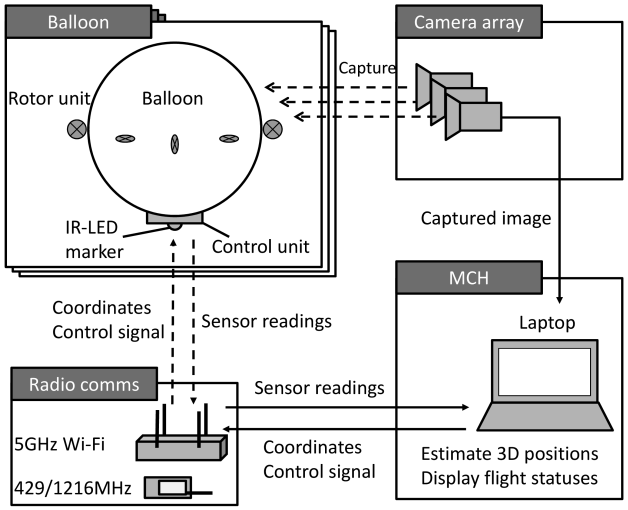

Fig. 7 System schematic

calculations. It is connected to the switch, therefore being able to retrieve the camera pictures at a rate of 6.6fps at 1920×1200/8bit per camera. Connected on the same switch a Wi-Fi router and the two of the same radio modules installed on the balloon are connected directly via USB.

3 Information flow and control

Fig. 7 gives a schematic overview over the data transmissions happening within the setup. The cameras record images, which are send to the MCH. There the locations of the balloons are calculated and the information is send to the respective balloons. With this knowledge of the interfaces between the components we will now go into the individual programs running.

3.1 Camera Tracking

Since the readings of the internal sensors and the history of velocity/force instructions are not enough to develop a suf-

Fig. 8 Raw camera image of an event venue. Note there are no IR markers captured in this figure.

Fig. 9 Cropped image of captured IR markers of two balloons. The diameters are approximately 20px.

ficient model of the balloon and with no way of correcting errors in such an approach it is crucial to have an external measurement to track and therefore correctly manipulate the balloons behavior.

As explained in section 2.2, we are capturing frames from six different angles at 6.6 fps. We are using OpenCV [6] as it provides fast and efficient algorithms to apply filters and other ways to manipulate image based data. As mentioned the IR light usually has low noise levels when indoors, however as there are usually at least some stationary IR emitters present (e.g., incandescent light bulbs and windows with daylight shining in). To counter this problem, we decided to create a mask for each camera, that will be used to filter out areas of high noise. Of course, this decreases the covered area, which however can be compensated for by having a larger number of cameras.

The image we receive from the cameras are already in gray scale due to only IR light passing the filter on the lens, so there is no need for additional processing.

Fig. 8 shows an IR image of the event site which a concert is being held. Some stational IR emitters (light bulbs and spotlights) can be seen in the figure. The multiple LEDs of the marker will be recognized as a single-continuous bright region (see Fig.9).

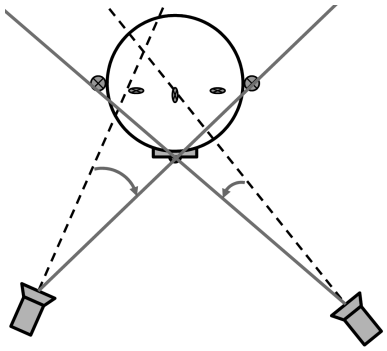

Fig. 10 Two camera cones extending 2 crossing vector lines

We can run functions provided by OpenCV to find contours of objects of a minimum size, filtering out any leftover noise and resulting in pixel ranges of the tracked object.

When flying multiple balloons, there will of course be multiple clusters of recognize pixels, however they can be differentiated and linked to a specific balloon based on proximity to the last known pixel cluster of each balloon. This way we only need to tell the system once, when deploying the balloons for the first time, where each marker is approximately located.

To translate the image data to a position in 3D space we must first know the angle of the image plain relative to the ground and the 3D position a defined set of points in image relate to. For this we use a replication of the IR marker located at the bottom of the balloon. By doing so we can figure out exactly what space the camera covers. It is a cone extending from the camera which gets cropped to become a pyramid to fit the 1920×1200 data we want to extract.

Now we can relate any pixel on the image to a unit vector extending from the camera's lens. Analyzing two or more of these vectors, performing a ray cast of and searching intersections will lead to a position in 3D space. Of course, these calculations are done, allowing a small error to account for imperfect calibration and noise. Many of these steps are provided by the OpenCV library. Fig. 10 shows a simplified example of this.

The average error of this positioning system is 26.7cm in a space of a gymnasium (36m $\times$ 24m $\times$ 7m).

Due to the tracking of a single marker attached on bottom of the balloon, we can't know the rotation of the balloon. To solve this problem, we are using the readings of the magnetic sensor included in the MPU-9250.

In addition, the readings of the MPU-9250 sensor is used as an inertial measurement unit (IMU) to compensate the suboptimal update speed of the camera and unexpected loss of wireless connection to the MCH. The algorithm used here is a basic double integral; therefore the cumulative error will increase over time.

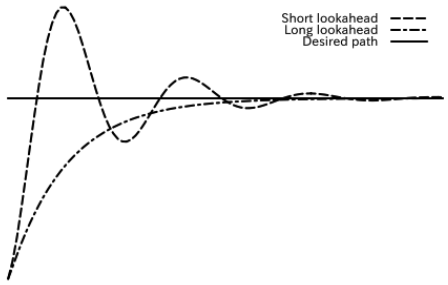

Fig. 11 Carrot chasing approach

3.2 Path following

To do effective path following there are a couple of options [7,8]. We chose to implement the carrot chasing approach [9]. First let us define a lookahead time t_L, which indicates how far in the future we try to exactly match the desired path. The choice of t_L is crucial, as a too short lookahead time will produce a system that can end up oscillating around the desired path. A too long lookahead time will result in the balloon taking very long to adjust to changes in direction and often having an offset to the path. Fig. 11 illustrates the basic idea behind this. Of course, this example only accounts for 1D straight line paths, but the general principle is the same for 2D or 3D non-straight paths, only with a much more noticeable error.

To understand the method itself, the choice of the lookahead time is arbitrary, as long as it is bigger than the sampling rate for discrete systems like ours. Therefore, let's assume we set $t_L = 0.5s$, together with our sampling rate of $\Delta t = 0.1s$ it is now obvious where the methods name comes from as we can define the desired velocity $\overrightarrow{v_d}$ as

$$\overrightarrow{v_d} = \frac{\overrightarrow{x}_d(t+t_L) - \overrightarrow{x_d}(t)}{t_L} \tag{1}$$

Lastly, from the wanted velocity we can get the force $\overrightarrow{F_d}$ in each direction and assuming a rigid setup we need to apply to the balloon via the basic laws of mechanical motion

$$\overrightarrow{F_d} = \frac{\overrightarrow{v_d}}{\Delta t} m \tag{2}$$

where we simplify the balloon's mass distribution to a point mass due to the symmetrical setup of the balloons. This vector is the output of the algorithm in section 3.3.

3.3 Balloon control

On the balloon, the Raspberry Pi functions as a mediator, translating the information it gets to the needed signals for the actuators on the balloon. This translation as several components in itself. The first input is the position of balloon received from the MCH and we calculate the force needed to change the movement of the balloon and stay on the desired trajectory as discussed in section 3.2.

For now, let's assume we are given the force we need to apply to follow the path, calculated from just the position and the velocity of the balloon. To apply the forces based on this signal, we first need to transform the coordinate system, from the one set by the cameras and the MCH $\overrightarrow{S_E} = (e_x, e_y, e_z)^T$ to the one of the balloon $\overrightarrow{S_B}$. With knowledge of the systems yaw position θ from the sensor readings, this can be done with:

$$S_B = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} S_E \tag{3}$$

A PID controller running for each of the forces we apply to influence the balloon $(f_{Bx}, f_{By}, f_{Bz}, f_\theta)$, where each f are the force toward the according direction in Fig. 2. Typically, the equation for a PID controller [10] is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \tag{4}$$

the proportional, integral and differential gains and $e(t)$ the error at time t. This however is most of the time implemented in the standard form:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{d}{dt} e(t) \right) \tag{5}$$

, where T_i and T_d refer to real physical parameters, namely the time in the future the errors are predicted and the time in the past errors are accounted for. However, since our system operates in discrete time steps a reformulation is possible and we obtain a much simpler algorithm defined by:

$$\begin{aligned} u(t) = u(t-1) + K_p \Bigg[& \left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t} \right) e(t) \\ & - \left(1 + 2\frac{T_d}{\Delta t} \right) e(t-1) + \frac{T_d}{\Delta t} e(t-2) \Bigg] \end{aligned} \tag{6}$$

with Δt the sampling time and in our case as we try to match the velocity accurately as possible $e(t) = v_d(t) - v(t)$ the error in velocity in the respective direction at time t. Note that this equation can easily be brought back to the general form by substituting $T_i = \frac{K_p}{K_i}$ and $T_d = \frac{K_d}{K_p}$.

As a final step, the instructions need to be split into signals for the individual rotors. Since their placement (Fig. 2) leads to them being dependent of each other, but the arrangement of the actuators is symmetrical and well-defined, the resulting transformation is given by:

$$\begin{cases} m_0(t) = -f_{Bx}(t) + f_\theta(t) \\ m_1(t) = +f_{Bz}(t) - f_\theta(t) \\ m_2(t) = -f_{Bx}(t) - f_\theta(t) \\ m_3(t) = +f_{Bz}(t) + f_\theta(t) \\ m_4(t) = m_5(t) = f_{By}(t) \end{cases} \tag{7}$$

where m_n are the thrust each rotor #n should output.

4 Results

As of the current version the balloon flies very consistent on the desired path, the average error is relatively small and it can handle different maneuvers. The desired path is shown in Fig. 12 and 13. This path was designed to test the balloon in fast and slow movements evenly. However, there are, as evident from the plot shown in Fig. 14. Parameter used during this flight is shown in Table. 1. Oscillations around the desired path similar to what one would suspect to happen if the lookahead time is too short. The root mean square(RMS) error of the position of the balloon is shown in Fig. 15. Average RMS error was 0.765m in the flight.

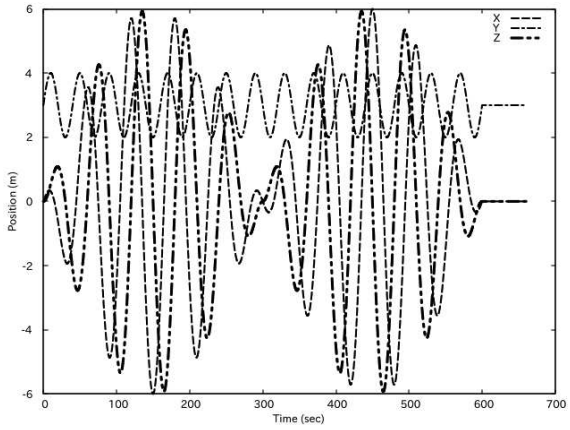

Fig. 12 Desired path

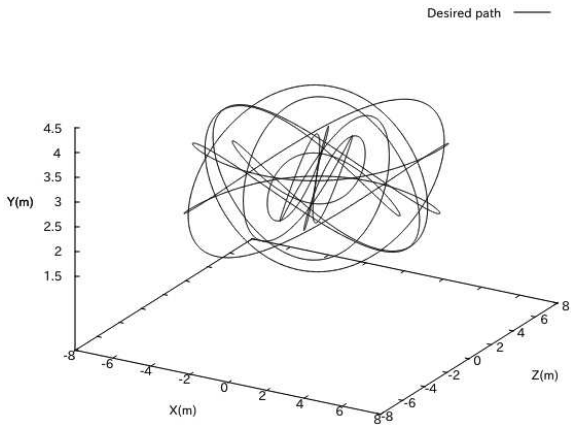

Fig. 13 Desired path shown in 3D

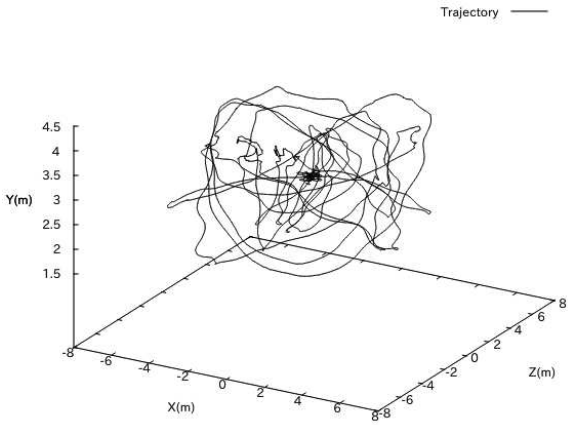

Fig. 14 Oscillation occurring during flight

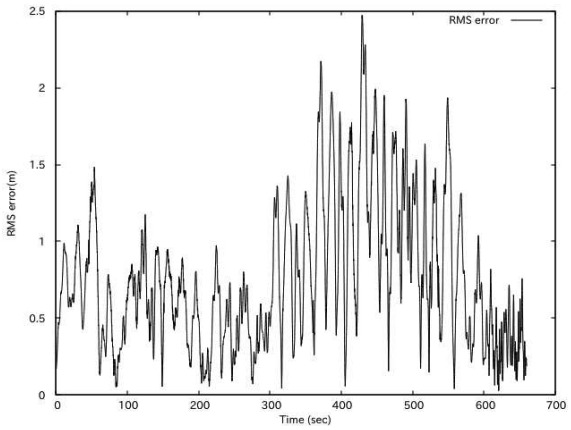

Fig. 15 RMS error of the position during the flight with oscillation occurring

However this is not the case, as we first encountered this problem after switching to a new balloon material. The center of mass of the balloon is following the desired path very accurately, but due to the new distribution of mass the system becomes more unstable and starts to develop roll and pitch motions which were unnoticeable before. Currently our setup is not build to handle such motions and mistakenly interprets them as unintended movement of the whole balloon in the x-y-z space instead of identifying changes in the two degrees of freedom that we previously did not have to deal with. We could limit the effects of this effect by adding more helium to the balloon and adding more weight to the bottom. The result can be seen in Fig. 16, 17 and Table. 2. Average RMS error was 0.505m, which is a 40% improvement in terms of RMS error which majorly caused by the oscillation.

Table 1 Parameters of oscillated flight

K_P	T_I	T_D	Extra weight
35	1/100	2	None

Table 2 Parameters of less oscillated flight

K_P	T_I	T_D	Extra weight
25	1/100	2	$\approx$ 200g

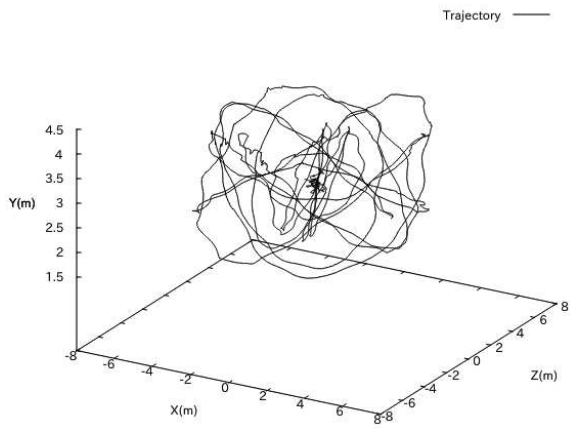

Fig. 16 Less oscillation occurring during flight

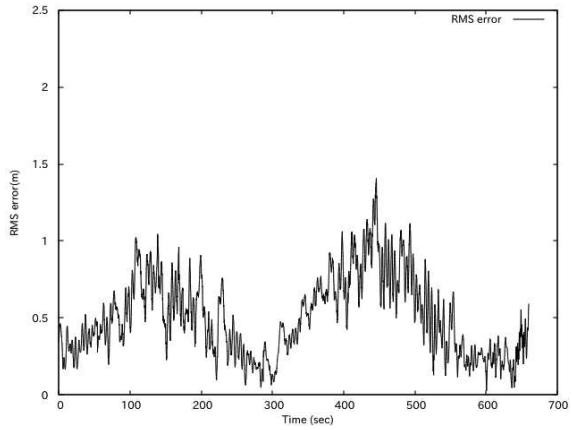

Fig. 17 RMS error of the position during the flight with less oscillation occurring

5 Conclusion

We could demonstrate that it is possible to create a balloon robot that can safely fly on set paths or follow waypoints and succeeded in manufacturing a prototype that has the build quality and durability to be of commercial use. However, for a system as sensitive as a floating balloon every decision is a tradeoff often with no ideal solution. As of right now we believe there are still improvements that can be made on the software side to minimize the mentioned problems:

- Further optimize the PID Gains to narrow down the error inducing effects
- Test neural network based PID controllers and research their applicability
- Improve the path following algorithm to a more complex and perhaps dynamic model that changes based on the path variables
- Filter out roll and pitch movements to improve tracking

Of course, some of the problems could also be approached by further optimizing the hardware setup:

- Add Actuators and controllers to stabilize roll and pitch
- Further experiment with materials and weight distribution.

All that said we think that the results that exist up until now are a great prove of concept and only need minor tweaking and optimizing to be able to satisfy our goal.

Acknowledgements A part of this research was supported by Nihon Stage Co., Ltd. Authors would like to thank Nihon Stage Co., Ltd. for their support.

References

1. H. Kawamura, H. Kadota, M. Yamamoto, T. Takaya, A. Ohuchi (2005) Development of an entertainment indoor blimp robot based on hovering control (in Japanese). J. of Japan Society for Fuzzy Theory and Intell. Informatics (17:203-211)
2. T. Takaya, H. Kawamura, Y. Minagawa, M. Yamamoto, A. Ohuchi (2006) PID landing orbit motion controller for an indoor blimp robot. Artif Life Robotics (10:177-184). doi:10.1007/s10015-006-0385-9
3. T. Takaya, H. Kawamura, Y. Minagawa, M. Yamamoto, A. Ohuchi (2008) Learning landing control of an indoor blimp robot for self-energy recharging. Artif Life Robotics (12:177-184). doi:10.1007/s10015-007-0451-y
4. S. van der Zwaan, A. Bernardino, J. Santos-Victor (2002) Visual station keeping for floating robots in unstructured environments. Robotics and Autonomous Systems (39:145-155). doi:10.1016/S0921-8890(02)00200-2
5. H. Kawamura, H. Iizuka, T. Takaya, A. Ohuchi (2009) Cooperative control of multiple neural networks for an indoor blimp robot. Artif Life Robotics (13:504-507). doi:10.1007/s10015-008-0604-7
6. OpenCV. http://opencv.org/. Accessed 19 February 2017
7. R. Cunha, C. Silvestre (2005) A 3D PATH-FOLLOWING VELOCITY-TRACKING CONTROLLER FOR AUTONOMOUS VEHICLES. IFAC Proc. Volumes (38:73-78). doi:10.3182/20050703-6-CZ-1902.02064
8. A. Pedro Aguiar, Joao P. Hespanha (2007) Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty. IEEE Transactions on Automatic Control (52:1362-1379). doi:10.1109/TAC.2007.902731
9. S. A. H. Tabatabaei, A. Yousefi-koma, M. Ayati and S. S. Mohtasebi (2015) Three dimensional fuzzy carrot-chasing path following algorithm for fixed-wing vehicles. 2015 3rd RSI International Conference on Robotics and Mechatronics (ICRoM) 784-788. doi:10.1109/ICRoM.2015.7367882
10. Kiam Heong Ang, G. Chong and Yun Li (2005) PID control system analysis, design, and technology. IEEE Transactions on Control Systems Technology (13:559-576). doi:10.1109/TCST.2005.847331