A Vision and GPS-Based Real-Time Trajectory Planning for a MAV in Unknown and Low-Sunlight Environments

Gerardo Ramon Flores Colunga, Shuting Zhuo, Rogelio Lozano, Pedro Castillo

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

A Vision and GPS-Based Real-Time Trajectory Planning for a MAV in Unknown and Low-Sunlight Environments

Gerardo Flores · Shuting Zhou · Rogelio Lozano · Pedro Castillo

Received: 02 Sep 2013 / Accepted: 10 Sep 2013

Abstract In this paper we address the problem of real-time optimal trajectory generation of a micro Air Vehicle (MAV) in unknown and low-sunlight environments. The MAV is required to navigate from an initial and outdoor position to a final position inside of a building. In order to achieve this goal, the MAV must estimate a window of the building. For this purpose, we develop a safe path planning method using the information provided by the GPS and a consumer depth camera. With the aim of developing a safe path planning with obstacle avoidance capabilities, a model predictive control approach is developed, which uses the environment information acquired by the navigation system. The results are tested on simulations and some preliminary experimental results are given. Our system's ability to identify and estimate a window model and the relative position w.r.t. the window is demonstrated through video sequences collected from the experimental platform.

Keywords MAV · RGB-D camera · Kinect · path planing

Gerardo Flores
Heudiasyc UMR 6599 Laboratory, University of Technology of Compiègne, France.
Tel.: +33 3 44 23 47 36
E-mail: gfloresc@hds.utc.fr

Shuting Zhou
Heudiasyc UMR 6599 Laboratory, University of Technology of Compiègne, France.
Tel.: +33 3 44 23 47 36
E-mail: shuting.zhou@hds.utc.fr

Rogelio Lozano
Heudiasyc UMR 6599 Laboratory, UTC CNRS France and LAFMIA UMI 3175, Cinvestav, México.
Tel.: +33 3 44 23 44 95
E-mail: rlozano@hds.utc.fr

Pedro Castillo
Heudiasyc UMR 6599 Laboratory, University of Technology of Compiègne, France.
Tel.: +33 3 44 23 44 84
E-mail: castillo@hds.utc.fr

1 Introduction

In the last few years there has been an ever-growing interest on development of micro air vehicles (MAV) due to its capabilities to fly in indoor/outdoor environments. MAVs can be used to explore terrains and acquire visual information in scenarios where it would be impossible for land vehicles. Additionally they are very suitable for various applications such as surveillance and reconnaissance operations, traffic monitoring or rescue missions in disaster sites, where manned or regular-sized aerial vehicles aren't able to accomplish such missions even with their fully operational capabilities.

To accomplish an efficient exploratory navigation in cluttered environments, the MAVs must be able to plan and follow three-dimensional trajectories and at the same time avoiding collisions with obstacles. Traditional navigation systems based on wirelessly transmitted information, such as Global Positioning System (GPS), are widely used to ensure a self-positioning task. However, most indoor environments remain inaccessible to external positioning systems, limiting the navigation ability of the satellite-based GPS systems.

Vision-based navigation arises as a complementary system for GPS. One of these navigation systems is the stereo vision system. Stereo vision has been popular mainly because they have the capability to include an amount of affordable information relative to the sensor expenses. Furthermore, texture and color information can be highly informative about drivable regions; also appearance and structural cues provide significant data about the image [1].

Based on stereo techniques, RGB-D cameras achieve better performance in the spatial density of depth data. Since RGB-D cameras illuminate a scene with a structured light pattern, they can estimate depth in areas with poor visual texture [2], thus the structured light RGB-D camera plays a supplementary role in the GPS-denied environment. However most RGB-D cameras operate in a limited range and they can't achieve a satisfactory navigation, especially when they are used as the only sensor. As a result, we combine GPS with an on-board RGB-D camera in order to provide the MAV with a fast and reliable state estimation and collision-free path planning.

There are previous studies conducted on the MAVs path planning with avoidance of collision. For instance, the Rapidly-exploring Random Tree (RRT) variant, which is proposed by Yang in [3]. To generate collision free piecewise paths, a Model Predictive Control (MPC) approach is applied. Yang has evaluated the robustness of the system by flying over, flying beneath or flying through obstacles, by using doors and windows of a building. Rasche and Stern in [4] applied an approach based on artificial potential fields and gradient method to calculate paths, which ensures the multiple UAVs complete a fast exploration of unknown, partially or completely known environments consisting of complex objects. For determining obstacles and objects in cluttered environments, a system for visual odometry and mapping using a RGB-D camera, is presented in [2]. Similarly, in [5] Henry developed a RGB-D mapping system that utilizes a novel joint optimization algorithm to generate dense 3D maps of indoor environment.

The RGB-D cameras have not generally been used in outdoor environments since they don't operate at intense sunlight [6]. However, recently there have emerged some works in which the Kinect sensor (which is a RGB-D camera) is used at outdoor environments. To cite some of these works; in [7] the author presents a RGB-D-based system which consists of a wheeled-robot equipped with a Kinect sensor. Such wheeled-robot is developed in a way such that it tracks some trails for autonomous outdoor navigation by using the Kinect. Rasmussen has demonstrated that the illumination conditions do not affect the performance of the RGB-D camera in scenarios where the sunlight is low enough. In [8] an example of the use of the Kinect sensor at outdoor environments is introduced. A Kinect was mounted on a bicycle, then the data was collected as the bicycle was ridden on streets. Planar patches were fit to the data and the normals were used to detect undrivable features such as poles or curbs.

In this work, we have chosen the Microsoft Kinect as the RGB-D sensor. The Kinect is a stereo camera which offers both color and depth information, i.e. RGB-D. The depth maps produced by the Kinect are highly accurate over a wide range of scenes due to it generates scene texture for stereo correspondence by active laser projection. The low price and its accuracy have made the Kinect a very compelling sensor. However, a limitation of the Kinect sensor is that sunlight interferes with the pattern-projecting laser, so it is most suitable for indoor applications. Nevertheless, outdoor applications are possible when sunlight is sufficiently diminished, i.e. in twilight, at night or at early morning conditions, with cloudy weather or even in strong shadow.

In this paper, we require that the MAV accomplishes the task of identifying a window and fly through it, in order to access into a building. The fulfillment of this objective will be quite significant for various military and civil missions. The present paper presents a solution to the real-time optimal trajectory generation of a MAV by integrating MPC and vision-based window estimation.

This paper is organized as follows. Section 2 addresses the problem of real-time trajectory planning. Section 3 presents the path-planning algorithm and obstacle avoidance method. Simulation results of the proposed path-planning algorithm are presented in Section 4. The vision-based window detection algorithm and some details about the experimental platform are introduced in Section 5. Finally, Section 6 draws a conclusion and gives a perspective on future work of the related research.

2 Problem statement

The first goal is to obtain a piecewise continuous function $u(t)$ that drives the MAV from a starting point x_0 to the final state x_f, using the optimal MPC approach. Figure (1) shows a three-dimensional example of an obstacle avoidance problem. As one can see in Fig. (1), the presence of obstacles in an arbitrary environment is inherent. Urban elements like lampposts, trees, light poles, power

cables and other civil buildings are represented as obstacles for the considered system. Hence, the state $x(t)$ of the dynamical system cannot take arbitrary values. Next, the problem statement is given: **Problem Statement** *Find a trajectory that allows the MAV to navigate from an arbitrary point* p_0 *with coordinates* $(x_0,\ y_0,\ z_0)$ *to a final point* p_f *with coordinates* (x_T, y_T, z_T, ψ_T)*, avoiding collisions with* n_o *obstacles in the environment. Once the MAV has achieved the point* p_f*, detect the window model an its parameters and minimize the distance between the centroid of the window model and the MAV center of gravity.*

Fig. 1 Scheme of the MAV application. The vehicle should compute a trajectory with obstacle avoidance capabilities using a vision system and GPS information.

2.1 Optimal control formulation

The equations representing the MAV motion can take different forms such as 1) nonlinear fully coupled, 2) nonlinear semi-coupled, 3) nonlinear decoupled, 4) linear coupled and 5) linear decoupled [9]. Due to the inherent load of any optimization process, we choose the linear decoupled version of the MAV dynamic model to generate the desired trajectory. The position dynamics of such model can be represented in its discrete form as follows

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + T\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}x(k) \end{aligned} \tag{1}$$

where $\mathbf{x} = (x, y, z)$ is the vector representing the MAV position in the inertial frame, $\mathbf{u} = (v_x, v_y, v_z)$ is the input control, $\mathbf{y}$ is the output vector, C is a matrix of appropriate dimensions and T is the sampling time.

Note that the linear and decoupled model (1) will be used by the optimization algorithm only to produce the desired trajectory. The trajectory block containing the path-following controller, will be responsible of following the generated trajectory, as shown in Fig. (2). We assume that the MAV is

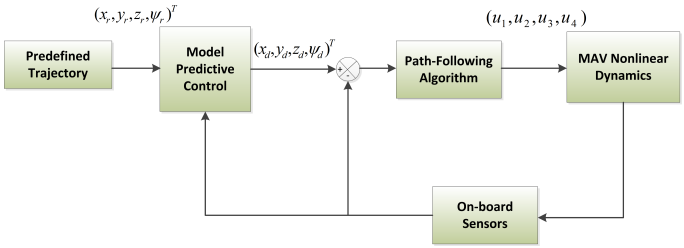


Fig. 2 Control scheme.

capable of receiving obstacle information at any path-planning trajectory by means of onboard sensors. In order to achieve the desired trajectory, we need to solve the following discrete-time optimal control problem:

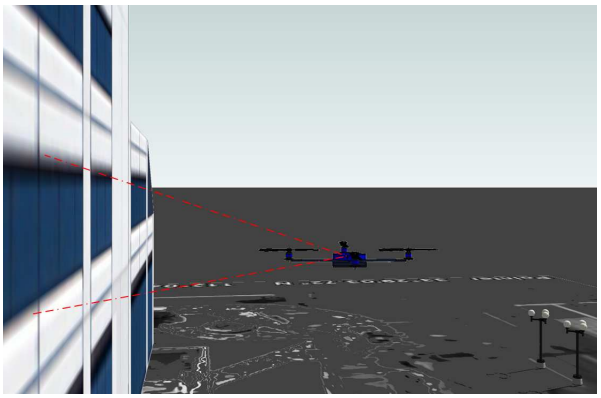


Fig. 3 Window model estimation. In this scenario the MAV has achieved the final point p_f provided by the GPS.

2.2 Optimal trajectory generation using MPC approach

Find the optimal sequence $\{u^*(k)\}_{k=1}^{T-1}$ *such that*

$$\{u^*(k)\}_{k=1}^{T-1} = \arg\min W(\mathbf{x}, \mathbf{u}, k) \tag{2}$$

where the positive definite cost function $W(\mathbf{x}, \mathbf{u}, k)$ is chosen as

$$W(\mathbf{x}, \mathbf{u}, k) = \Phi\left(\mathbf{x}(T)\right) + \sum_{k=1}^{T-1} \mathfrak{L}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \tag{3}$$

subject to system (1). In order to find the optimal control law (2) for system (1), we propose a nonlinear model predictive control approach. We proceed by starting from the initial state $\mathbf{x}(1)$ and then implementing the optimal input $\{u^*(k)\}_{k=1}^{T-1}$ from $1 \leq \tau \leq T$ to the state $\mathbf{x}(\tau+1)$ at $k = \tau + 1$. The key idea behind the MPC approach is the combination of the potential field concept with the online optimization with preview. In this way, the function $W(\mathbf{x}, \mathbf{u}, k)$ is minimized by the optimal control sequence (2), i.e.

$$W(\mathbf{x}, \mathbf{u}^*, k) \leq W(\mathbf{x}, \mathbf{u}, k), \qquad \forall u \in U \tag{4}$$

3 Real-Time trajectory planning

In this section, we present a trajectory generation algorithm for the task described in the previous section.

3.1 Trajectory planning

Consider the navigation problem starting from the initial point p_0 and reaching to the final point p_f, where p_0 is the MAV's initial position given by the GPS and p_f is a point near to the window model to be estimated, which is provided by the GPS (Fig. 3). In order to achieve the desired position in the presence of obstacles, the procedure developed in Section 2.2 is applied. We consider an initial reference trajectory given by a straight line. Note that any other reference trajectory can be used. Consider a cost function term for (3)-(2) as

$$\begin{aligned} \Phi\left(\mathbf{x}(T)\right) &\triangleq \frac{1}{2}\mathbf{x}^T(T) S \mathbf{x}(T) \\ \mathfrak{L}\left(\mathbf{x}(k), \mathbf{u}(k)\right) &\triangleq \mathfrak{L}^t\left(\mathbf{x}(k), \mathbf{u}(k)\right) + \mathfrak{L}^o\left(\mathbf{x}(k), \mathbf{u}(k)\right) \end{aligned} \tag{5}$$

where

$$\begin{aligned} \mathfrak{L}^t\left(\mathbf{x}(k), \mathbf{u}(k)\right) = &\frac{1}{2}\left(\mathbf{x}_r - \mathbf{x}\right)^T Q \left(\mathbf{x}_r - \mathbf{x}\right) \\ &+ \frac{1}{2}\mathbf{u}^T R \mathbf{u} \end{aligned} \tag{6}$$

3.2 Obstacle sensing

For collision avoidance, the function $\mathfrak{L}^o\left(\mathbf{x}(k), \mathbf{u}(k)\right)$ is chosen in (5) such that

$$\mathfrak{L}^o\left(\mathbf{x}(k), \mathbf{u}(k)\right) = \sum_{i=1}^{n_o} f(\mathbf{x}, \mathbf{x}_i). \tag{7}$$

The function $f(\mathbf{x}, \mathbf{x}_i)$ is defined as

$$\begin{aligned} f(\mathbf{x}, \mathbf{x}_i) = a_x \exp\left(-\frac{(\mathbf{x}(k) - \mathbf{x}_i(k))^2}{2c^2}\right) \\ + a_y \exp\left(-\frac{(\mathbf{y}(k) - \mathbf{y}_i(k))^2}{2c^2}\right) \\ + a_z \exp\left(-\frac{(\mathbf{z}(k) - \mathbf{z}_i(k))^2}{2c^2}\right) \end{aligned} \tag{8}$$

where $(\mathbf{x}_i,\ \mathbf{y}_i,\ \mathbf{z}_i)$ are the coordinates of the i-th obstacle in the inertial frame. The parameters a_x, a_y, a_y, c are chosen to determine how far the MAV can approach the obstacle. Thus, the penalty function (8) serves as a repelling field.

4 Simulation results

In order to test the effectiveness of the derived controller, in this section some simulations results are given. The aforementioned optimization procedure will result in the desired trajectory which will be tracked by the path-following algorithm, see Fig. 2.

We apply the optimization procedure for the navigation problem in which the MAV is requested to fly from the initial point $p_0 = (0, 0, 0)$ to the final point $p_f = (10, 10, 10)$. We have simulated the presence of an obstacle in the coordinates $(\mathbf{x}_i,\ \mathbf{y}_i,\ \mathbf{z}_i) = (5, 5, 5)$. This scenario often arises in urban areas, as can be seen in Fig. 1.

The collision-avoidance problem is solved by maintaining a safe distance from the nearest point of nearby obstacles throughout the flight envelope. Fig. (4) shows the output trajectory. The velocities generated by the algorithm are depicted in Fig. (5). Such velocities and the corresponding positions are used in the feedback control loop as a reference to the position control loop. By choosing different values of the parameter c, it is possible to avoid obstacles from a longer distance as shown in Fig. 6. (6).

5 Vision-based window estimation

In order to solve the second part of the problem statement presented in Section 2, a vision-based algorithm is proposed in this section. At each pixel of

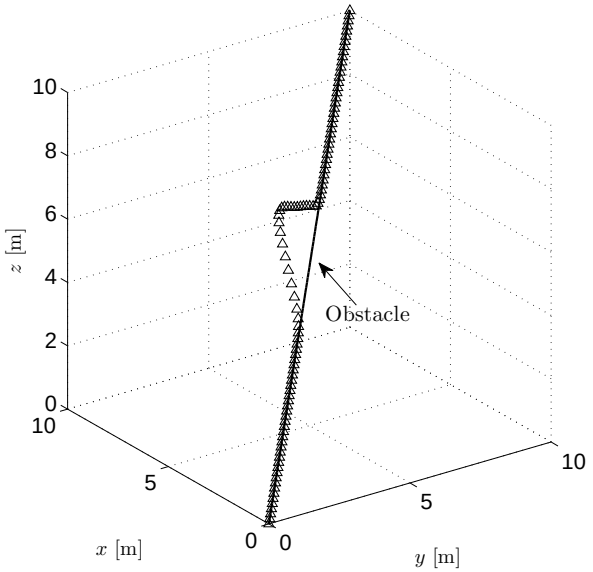


Fig. 4 Computed path. An obstacle is positioned at the coordinate $(5, 5, 5)$.

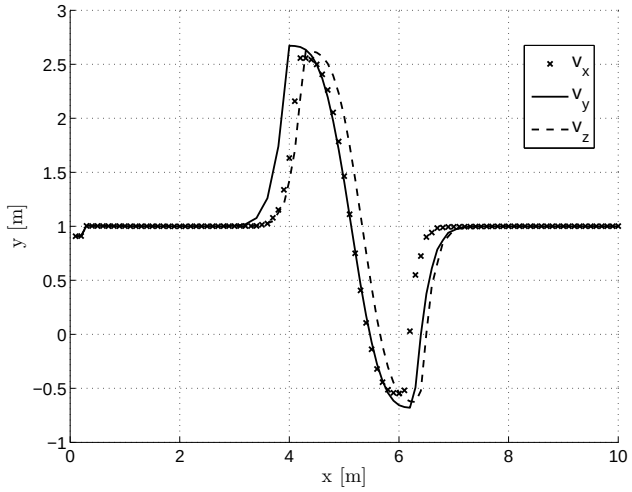


Fig. 5 Velocities generated by the algorithm.

the image, RGB color information and depth data are acquired by the RGB-D camera. Such kind of cameras manage all the information with a set of vertices in a three-dimensional coordinate system called *point cloud*. Point cloud is a collection of massive points in the same spatial reference system, which expresses the spatial distribution and the surface properties of a target, which

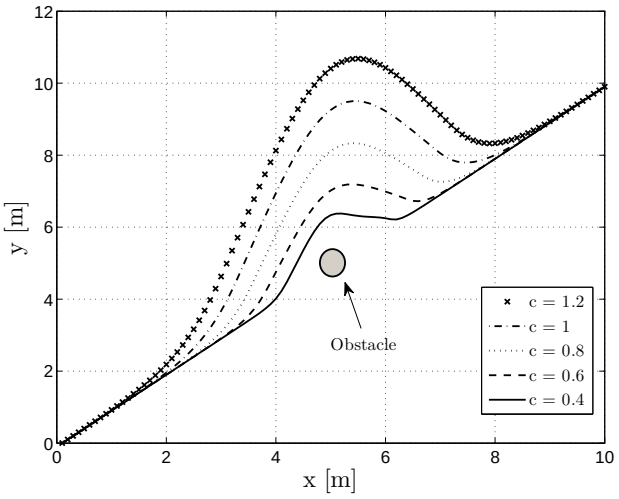


Fig. 6 Trajectories.

in this case is the window model [10]. Three-dimensional coordinates of each sample point on the target surface (x, y, z) and the corresponding color information can be acquired from the point cloud. By using this information, an algorithm responsible for obtaining an estimate of the window model position with respect to the MAV is proposed below.

5.1 Algorithm description

The algorithm executes a down-sampling process on the input point cloud to improve the running efficiency. Then the iterative estimation process is performed to extract the inliers of planes existing in the point cloud until the target plane is detected. The approach applied for plane detection is the RANdom SAmple Consensus (RANSAC) algorithm. The basic idea of RANSAC is to estimate the plane parameters using the minimum number of possible data (random three points) and then, check which of the remaining data points fit the estimated model [11].

Two planes are given in the input point cloud, the larger one is the plane of the wall, while the smaller one is the plane representing the surface of the window model, which we are interested in. Procedure 1 in Fig. (7) describes the target plane identification process. With the estimated target plane, we continue to extract some key points representing the critical features in order to estimate the centroid of the target surface. Procedure 2, shown in Fig. (7) and explained in Tab. 1, describes how the key points are extracted from the point cloud of the estimated plane. Here we applied the Normal Aligned Radial Feature method (NARF) for interest point detection and feature descriptor calculation in 3D range data [12]. As NARF method takes information about

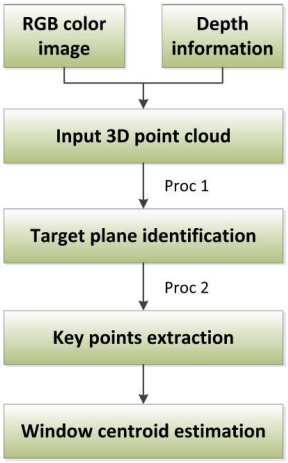


Fig. 7 Window identification process scheme.

Procedure 1 EstimateTargetPlane (PC: input point cloud, n: number of points in PC)

```
PC_d = DownSample(PC)
cloud_filtered = PC_d
i = 0
while size_cloud_filtered >n*8% do
   plane_i = ExtractInliers(cloud_filtered)
   cloud_filtered = cloud_filtered - plane_i
   i++
end while
return plane_1
```

Table 1 Procedure 1: Estimation of target plane.

Procedure 2 ExtractKeyPoints (PC: point cloud of plane)

```
range_image = CreateRangeImage(PC)
points[] = DetectNarfKeypoints(range_image)
return points[]
```

Table 2 Procedure 2: Extracting key points

borders and the surface structure, the position of the centroid can be estimated by using the edge information extracted by the NARF detection algorithm. To identify the window model and estimate its centroid, we take advantage of the programs available in the open source point cloud library [13]. Some preliminary results are presented next.

5.2 Experimental results

In order to perform the real-time implementation of the proposed strategy, we use the Microsoft Kinect sensor [6]. As a low-cost RGB-D sensor developed

by PrimeSense®, Kinect is equipped with three lenses, the lens in the middle corresponds to the RGB color camera, while the lenses in the left and right side are the infrared transmitter and infrared CMOS camera which constitute a 3D structured light depth sensor (Fig. 8).

Based on the light coding, Kinect projects a known infrared pattern onto the scene and determines depth based on the pattern's deformation captured by the infrared CMOS imager [14]. Functioning in this way, Kinect can provide a 320x240 depth image at 30fps and a 640x480 RGB color image at 30fps. When stripped down to its essential components, the Kinect weighs 115g, light enough to be carried by a MAV. In order to verify the algorithm, we

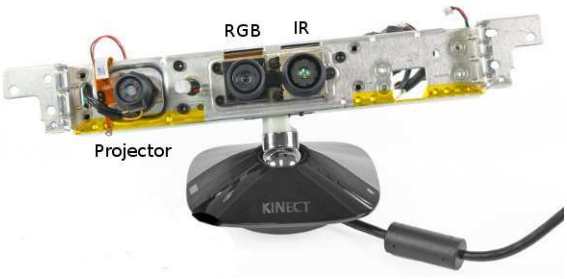


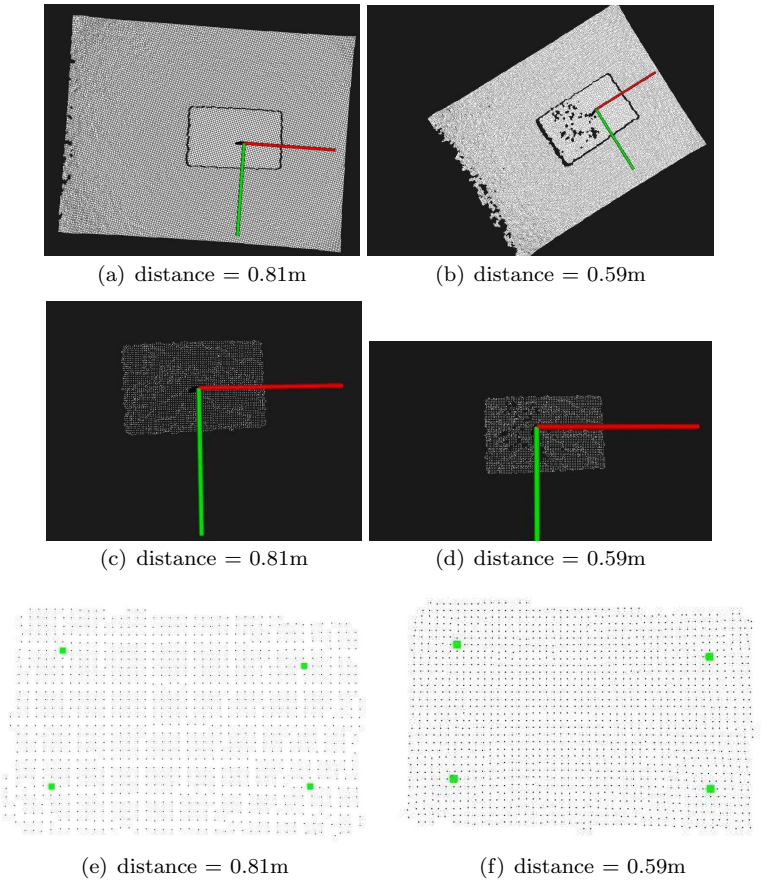
Fig. 8 The Microsoft Kinect without the cover [15].

have assumed that the window model is just in front of the Kinect sensor (Fig. 3). Thus, placed directly opposite to the target surface of the window model, the RGB-D camera collects sufficient position information of the target plane.

Several test have been performed, and the corresponding images are depicted in Fig. 9. The distance between the RGB-D camera and the window model is chosen to be 0.92m. Fig. 9a-b shows the input 3D point cloud captured by RGB-D camera. From the figure we can clearly recognize the surfaces of the wall and the window model surface. Fig. 9c-d shows the point cloud of the plane detected by the target plane identification algorithm. The target surface of the window model is completely extracted from the input point cloud. Fig. 9e-f shows four key points extracted from the range image of target plane. Such key points can be approximately regarded as the four vertices of the target plane. Then the centroid of the target plane can be obtained by intersecting the diagonals of the four key points. It is noteworthy that the execution time of the vision-based window estimation algorithm is about 40ms.

5.3 Quad-rotor experimental MAV

The quad-rotor developed on the HEUDIASYC laboratory is shown in Figure 10. It has been built using a group of commercially available components. The body frame is an MK-40 from MikroKopter. The distance between same



(a) distance = 0.81m (b) distance = 0.59m

(c) distance = 0.81m (d) distance = 0.59m

(e) distance = 0.81m (f) distance = 0.59m

Fig. 9 Target plane at closer distances. Figures (a)-(b) show the input point cloud. Figures (c)-(d) show the detected target plane. Figures (e)-(f) show the extraction of the four key points.

axis rotors is 40cm. Motors are BL-Outrunner from Robbe ROXXY, which are driven by BlCtrl I2C electronic speed controllers. The weight of the rotorcraft is 1.1 kg. It has a 11.1V - 6000 mAh LiPo battery, allowing an autonomy of about 15 minutes. The onboard electronics are based on the IGEPv2 card, equipped with a Texas Instruments DM3730 System On Chip (SOC). The SOC benefits from having an ARM CortexA8 core running at 1 GHz, and a C64x+ DSP core running at 800 MHz. The ARM core allows execution of Linux, as well as its real-time extension Xenomai. The quad-rotor sensor suit consists of the next group of components. Inertial measurements are provided at 100 Hz by means of a 3DMGX3-25 IMU from Microstrain®. A SRF10 ultrasonic range finder provides the vehicle altitude at 50 Hz in a range between 0 m and 2 m. All the previously mentioned components are fully embedded onboard the vehicle.

Fig. 10 MAV experimental platform: The quad-rotor equipped with imaging, inertial and altitude sensing systems.

6 Concluding remarks

In this work we have presented a new approach to the real-time optimal trajectory generation of a MAV by integrating MPC and vision-based window estimation algorithm. We have also extended some preliminary experimental results to verify the vision algorithm. Future work include implementing the vision-based navigation control strategy, onboard the experimental platform.

References

1. R. Manduchi, A. Castano, A. Talukder, L. Matthies, Autonomous Robots **18**, 81 (2004)
2. A.S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, N. Roy, in *Int. Symposium on Robotics Research (ISRR)* (Flagstaff Arizona, USA, 2011)
3. K. Yang, S. Gan, S. Sukkarieh, Journal of Intelligent and Robotic Systems **57**(1-4), 101 (2010)
4. C. Rasche, C. Stern, L. Kleinjohann, B. Kleinjohann, in *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on* (Wellington, New Zealand, 2011), pp. 7–12
5. P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, The International Journal of Robotics Research **31**(5), 647 (2012)
6. Microsoft, in *Internet* (`http://www.xbox.com/en-US/kinect`, 2013)
7. C. Rasmussen, in *RGB-D: Advanced Reasoning with Depth Cameras in conjunction with RSS 2012* (Sydney, Aus, Jul, 2012)
8. A. Robledo, S. Cossell, J. Guivant, in *Australasian Conference on Robotics and Automation* (Melbourne, Aus, Dec, 2011)
9. M. Sadraey, R. Colgren, in *AIAA Modeling and Simulation Technologies Conference and Exhibit* (San Francisco, California, 2005)
10. P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, in *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS* (2010)
11. F. M., R. Bolles, Communications of the ACM 24 p. 381395 (1981)
12. B. Steder, R. Rusu, K. Konolige, W. Burgard, in *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (Taipei, Taiwan, Oct, 2010)
13. R.B. Rusu, S. Cousins, in *IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, 2011)

14. X. Xiang, Z. Pan, J. Tong, Journal of Frontiers of Computer Science and Technology (2011)
15. ROS.org, in *Internet* (`http://wiki.ros.org/kinect_calibration/technical`, 2012)