2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Visual 3D Self Localization with 8 Gram Circuit Board for Very Compact and Fully Autonomous Unmanned Aerial Vehicles

Ryo Konomura[1] and Koichi Hori[2]

Abstract— **We describe an algorithm and hardware system of a 3D self-localization method for very cost effective hovering robots. The hardware system consists of very compact calculation and sensor units, instead of using a high performance calculation unit like a laptop PC. In the experiment, the comparison with the ground truth sensor is discussed and implementation of the system on a palm-sized quad-copter is studied to realize a very compact, on-board, fully autonomous quad-copter.**

I. Introduction

Small UAVs (unmanned aerial vehicles) play an important role in many fields due to their capability to move around indoors and outdoors with various sensors on their bodies. They can be used for a wide range of practical applications such as surveillance, rescue missions and entertainment.

Because wheel odometry based positioning technologies cannot be used in UAVs, most of the systems depend on visual systems to know their positions in a GPS-denied environment.

Using motion capture systems is one of the most popular methods[2][3][4] to determine position and very accurate performances are reported in those studies, though the usable space is limited to its visual range. And using a laser scanner is an effective solution for larger hovering robots[5][6].

The other method is to use monocular or multiple cameras implemented on the UAV using the VSLAM(Visual SLAM) method. Some of the systems depend on ground station computers to calculate and estimate the self-position[7][8][9], and fully integrated on-board systems have been studied since a few years ago[10][11], where the on-board computer performs self-position, attitude estimation, map reprojection and hovering control.

Those systems can achieve practical applications in a GPS-denied environment using fully or semi-autonomous navigation. But the body size, sound, and energy consumption are problems that still need to be solved for small indoor environments such as residential houses. For practical use in such an environment, a much smaller system is required, and this is our motivation for this study.

In the related studies of navigation systems of hovering robots with monocular cameras, PTAM(parallel tracking and mapping)[14], which is one of the VSLAM methods, is widely applied to ground station computers[7][8][9] and on-board computers[10][11]. The native PTAM method estimates camera pose and provides a 3D points map using a single hand-held camera and a laptop computer in a small AR (augmented reality) work-space. This method performs visual tracking and mapping in two separate parallel threads; thus, more than one core CPU is required. The library is open source, and programs can be compiled and installed in laptop or desktop computers with installed OSs.

TABLE I: Hardware Comparison

	Related systems	Our system
CPU	2.66GHz[14] 1.6GHz[10][11]	166MHz
RAM	No mention[14] 1.0GB[10][11]	128KB

Table I shows a comparison between their hardware systems and our hardware systems. Their systems use a GHz-order CPU with Gbyte-order RAM, mainly because PTAM requires large computational resources to track and map thousands of feature points to estimate self-position and a 3D map. And there is a problem that other processes required for practical application (e.g. recognizing objects based on machine learning and calculating complicated action plans based on other sensor information) may be hard to implement with the heavy computational burden of self-position localization and mapping.

In contrast, our study proposes a more light-weight 3D self-localization method that can work with quite less resource. Our computational system consists of a very low cost MCU (micro control unit) , FPGA (field-programmable gate array) , IMUs (inertial measurement units), and CMOS camera. The maximum frequency of the system is 166 MHz, and the total memory size is 128 KB, and its total weight is only 8 grams as shown in Fig. 2.

There are related studies applying a low cost MCU to micro helicopter robots[12][13]. However, those systems can only measure self-velocity using an optical flow algorithm but cannot perform absolute position localization.

This paper has two main topics. The first is a 3D self-position localization method using a monocular camera, MCU and FPGA. The second is the implementation of our computational system on a palm-sized hovering robot whose mechanical framework and electronic circuits were designed in our previous study[1] as shown in Fig.1, and the purpose is to show our study contributes to significant downsizing of on-board and fully autonomous hovering robots.

[1]Department of aeronautics and astronautics, the University of Tokyo. konomura at ailab.t.u-tokyo.ac.jp
[2]Department of aeronautics and astronautics, the University of Tokyo. hori at computer.org

Fig. 1: We have built Palm-sized Quad-copter

II. System Overview

Before explaining our proposal, two main limitations of our system compared to other studies using PTAM should be listed.

- Fewer than 200 feature points per image are used due to memory limitation.
- Resized image pyramids are not used for streaming calculation without storing "raw image".

But we show that despite the decrease in the number of points, good 3D self-position estimation can be obtained especially in the planar scene, and the comparison with the data from a ground truth sensor is described in the Implementation and Experiments section.

The overview of our system is shown in Fig.2. It shows the connection of each components and data flow processes. The following functions are described later.

1) Feature points detection and description (in the FPGA)
2) Calculation of self-attitude by IMU sensor fusion (in the MCU, periodically called function)
3) Determining corresponding feature points, and performing storage management and bundle adjustment (in the MCU, loop function)

First, image data from the CMOS camera (160x120 YUV422 format color image) is streamed to the FPGA. The FPGA performs feature point detection (FAST[15]) and description (BRIEF[16]). Each piece of feature point includes the location in the 2D image plane and 32-byte BRIEF descriptors. They are sent to 8 KB of MCU SRAM via DMA (direct memory access) bus. This 8 KB of memory is treated as a "ring buffer" and this data transaction occurs immediately if a new feature point is found in the FPGA process. When the processing of one image is finished, the FPGA sends a signal to the MCU to notify it that all image descriptors transactions are finished.

The MCU has two main tasks. The first is to calculate self-attitude using a sensor fusion algorithm of 9-DOF (degrees of freedom) sensors (3-axis accelerometer, 3-axis gyro, and 3-axis magnetometer), and that information is used not only for image processing but also for hovering control of the quad-copter. The second is to calculate 3D self-position from feature points information sent by the FPGA. This

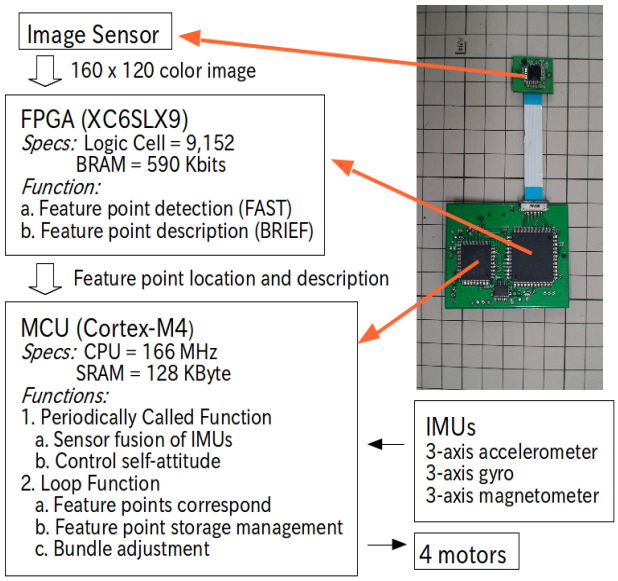

Fig. 2: Circuit System Overview

procedure mainly consists of three processes; determining corresponding feature points, managing memory for newly appearing and disappearing feature points, and performing bundle adjustment.

The MCU has two functions; loop function and periodically called function. The periodically called function is called every 10 ms and the loop function is called when the periodically function is not called. And the MCU tasks explained above are stated in either of these functions.

Because of space limitations, we have omitted the explanation of the control method of quad-copter to follow a route, which we would like to present in another paper.

III. Image Processing on FPGA

We propose an FPGA-based feature points tracking algorithm, which we named "CSFB"(Concurrently Streamed FAST and BRIEF), which has the following capabilities.

1) Can operate in real time (at least 30fps).
2) Tracking scale and rotation movements is possible for a real time operation.
3) No external memories to save images are required.
4) No DSPs (digital signal processor) for multiplication and division operations are required.

The details of implementation of this algorithm on a low-cost FPGA are as follows.

In our hardware system, an image sensor is implemented and sends a 160 x 120 ($hpix = 160$, $vpix = 120$) YUV422 format color image to the FPGA at 30Hz.

We have implemented the feature point detection (FAST corner detector[15]) and description (BRIEF[16]) engine on the FPGA. External storage devices to save the whole image are not needed because the FPGA can perform the above process at the same speed as the updating of the new image by the image sensor.

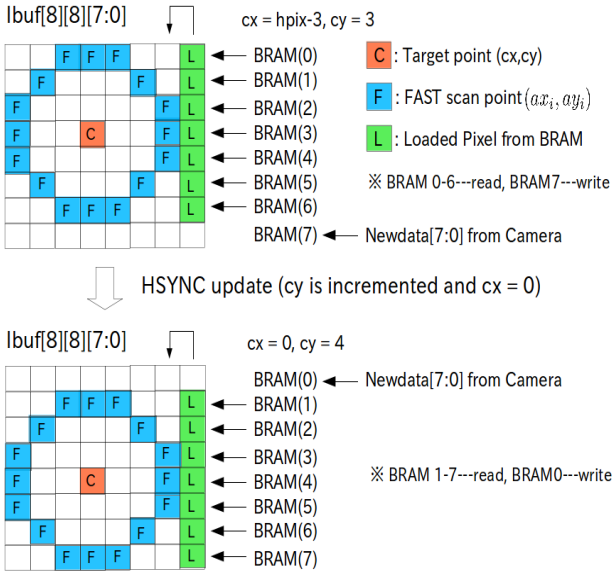

Fig. 3: Buffer Scheduling For Image Processing

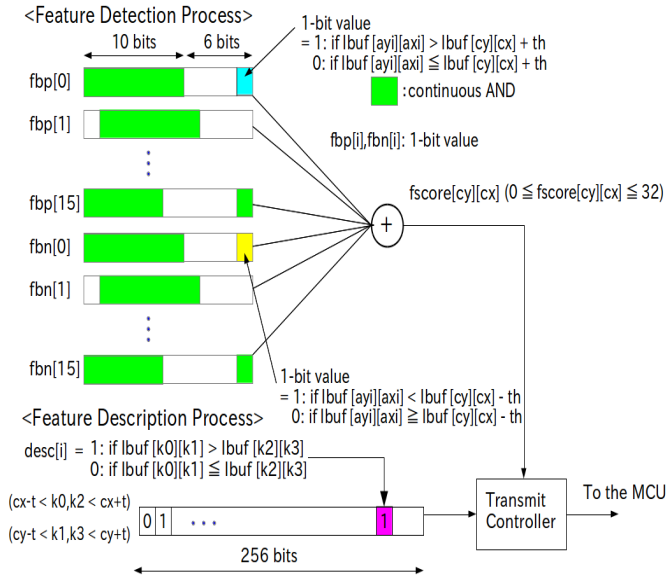

Fig. 4: Feature Detection and Description

The FAST corner detector classifies a pixel as a corner if n continuous circular pixels around that pixel is all brighter or darker than a threshold (FAST-n).

The BRIEF descriptor is a binary string generated by comparing the intensity of the target pixel with that of a pixel near the target point chosen under an established rule. BRIEF-m means m bytes of the binary string generated by performing pixel intensity comparison $m \times 8$ times.

Both FAST and BRIEF algorithms consist of a simple formula with much iteration, so the FPGA is very suitable for solving the problem quickly by parallelizing the repeated computation.

We used brightness data of 8 bits per pixel for each image process ($Newdata[7:0]$ in Fig. 3), and new pixel information arrives every two clocks of the camera clock. Therefore, the image process must be performed within the period for this update.

We designed the FPGA process to satisfy the above requirement, and its overview is shown in Fig.3. The $8\times8\times8$ bits of image buffer ($Ibuf(l)[j]$, $0 \le l \le 7$ and $0 \le j \le 7$) are reserved. And 8-Block RAMs with 8-bit data bus length are used ($BRAM(l)[m]$, $0 \le l \le 7$ and $3 \le m \le hpix - 3$). One of the 8-block RAMs is used to write the data, and the others are used to read the data, and their roles change in accordance with the count of the horizontal synchronization (HSYNC) rising signal as shown in Fig. 3.

Every two clocks of the camera clock, the following process is performed.

$$BRAM\,(l)\,[m] = Newdata[7:0], \quad (1)$$
$$Ibuf[!l][7] = BRAM(!l)[m], \quad (2)$$
$$Ibuf[!l][j] = Ibuf[!l][j+1], \quad (3)$$
$$(3 \le n < vpix - 3,\ 3 \le m < hpix - 3,\ l = n\%8,\ 0 \le j \le 6)$$

where !l means "not l" and $\%8$ means reminder divided by 8.

Next, image processes using the center pixel ($Ibuf[cy][cx]$) are performed as shown in Fig. 4. In our system, feature detection and feature description processes run simultaneously by parallelizing and multi-staging small calculations such as bit shift and comparison of two values, and finishing each pixel calculation within two clocks.

In the feature detection process, though native FAST closes out searching for ambient pixels based on a machine learning algorithm, our feature detection process tries all patterns to be recognized as a corner point when FAST-10 algorithm is applied (32 patterns are shown in Fig. 4 as $fbp[i], fbn[i]$, where $0 \le i \le 15$) And the score of corner point can be calculated by summing the 1-bit output of each corner pattern detector ($fscore[cy][cx]$) which are used to eliminate "weak" feature points near a "strong" corner point.

Each corner detector evaluates the brightness or darkness of pixels of 16 points around the center pixel ("F" in Fig.3) by constant threshold (th) and calculates the feature point score ($fscore[cy][cx]$) as follows.

$$fbp[i] = Ibuf[ay_i][ax_i] > (Ibuf[cy][cx] + th)\ ?\ 1 : 0, \quad (4)$$
$$fbn[i] = Ibuf[ay_i][ax_i] < (Ibuf[cy][cx] - th)\ ?\ 1 : 0, \quad (5)$$
$$fep[i] = fbp[i\%16]\ \&\ ...\ \&\ fbp[(i+9)\%16], \quad (6)$$
$$fen[i] = fbn[i\%16]\ \&\ ...\ \&\ fbn[(i+9)\%16], \quad (7)$$
$$fscore[cy][cx] = \sum_{i=0}^{15} fep[i] + \sum_{i=0}^{15} fen[i]. \quad (8)$$

In the feature description process, 256 pairs of 2-point pixels near the center points are extracted and 256-bit binary vectors are generated in accordance with the following equation (BRIEF-32). We tried several patterns of chosen pairs by randomly generating $k_0^i, k_1^i, k_2^i, k_3^i$ patterns to find

out good combinations of pairs.

$$desc[i] = Ibuf[k_0^i][k_1^i] > Ibuf[k_2^i][k_3^i] \ ? \ 1 : 0. \quad (9)$$
$$\left(cy - 3 \leq k_0^i, k_2^i \leq cy + 3, \ cx - 3 \leq k_1^i, k_3^i \leq cx + 3\right)$$

Then, the 256-bits binary vector is passed to the transmission process if the target point is recognized as a corner point by the feature detection process, that is, if $fscore[cy][cx]$ is more than zero. Each piece of feature point information includes 2 bytes of feature point location, 32bytes of descriptors and, 1 index byte, which counts the number of vertical synchronization signals. This counter enables the MCU to know whether the FPGA obtains all descriptors of the image. And those data are continuously sent to the MCU using the DMA (direct memory access) bus so that the MCU can get descriptor information asynchronously and does not need to consume CPU resource.

IV. MCU Processes

A. Corresponding feature points

The MCU corresponds the feature points by connecting feature point information of the current and previous image frames. This process is performed by simply calculating the XOR operation of two BRIEF-32 descriptors followed by a bit count. If this count is low, these two feature points have a high possibility of being the same point in the current and previous frame.

The BRIEF descriptor itself is not invariant to scale and rotation change of image. But by updating these descriptors after the corresponding process in a short interval, it is possible to successfully perform the corresponding process for any camera movements, based on the assumption that the camera does not make any large movements in the image update period. And that assumption enables the MCU to drastically reduce the search region of corresponding points around the target point.

And the 30fps update rate yields corresponding results that are good enough for normal hand movement and fully autonomous hovering as can be seen in our attached video.

B. Sensor Fusion

The MCU calculates its attitude from IMUs in a periodically called function. We use the quaternion-based gradient descent sensor fusion algorithm proposed by Madgwick et al.[17]. They define the following objective function when 3-axis accelerometer measurement ${}^S\hat{a}$ is used,

$$f\left({}^S_E\hat{q}, {}^E\hat{d}, {}^S\hat{a}\right) = {}^S_E\hat{q}^* \otimes {}^E\hat{d} \otimes {}^S_E\hat{q} - {}^S\hat{a}, \quad (10)$$

where ${}^S_E\hat{q}$ is the orientation of the sensor, ${}^E\hat{d}$ is the reference direction of the earth frame, and ${}^S_E\hat{q}^* \otimes {}^E\hat{d} \otimes {}^S_E\hat{q}$ with quaternion compound product $\otimes$ means the reference direction relative to the body frame.

Then, ${}^S_E\hat{q}_{\nabla,t}$ is calculated by the following gradient descent equation so that f is minimized.

$${}^S_E\hat{q}_{\nabla,t} = {}^S_E\hat{q}_{est,t-1} - \mu \frac{\nabla f}{\| \nabla f \|}, \quad (11)$$

where ${}^S_E\hat{q}_{est,t-1}$ is the previously estimated orientation of the sensor frame relative to the earth frame. After that, the current estimated orientation ${}^S_E\hat{q}_{est,t}$ is calculated by the following sensor fusion equation with the 3-axis gyro sensor

$${}^S_E\hat{q}_{est,t} = \gamma {}^S_E\hat{q}_{\nabla,t} + (1 - \gamma) {}^S_E\hat{q}_{\omega,t}, \quad (12)$$

where ${}^S_E\hat{q}_{\omega,t}$ is the orientation estimated by using the 3-axis gyro sensors and γ is the weight coefficient to decide which sensor's result should be placed much value. ${}^S_E\hat{q}_{\omega,t}$ is given by the following basic quaternion operation,

$${}^S_E\hat{q}_{\omega,t} = {}^S_E\hat{q}_{est,t-1} + \frac{1}{2} {}^S_E\hat{q}_{est,t-1} \otimes {}^S\omega_t \times \triangle t, \quad (13)$$

where ${}^S\omega_t$ is the 3 axis gyro sensor measurement and $\triangle t$ means the update intervals.

The 3-axis magnetometer is also applied to these forms of equations to calculate yaw angle though this is not explained in detail in this paper.

C. Storage managements

Now the MCU has corresponding information of each feature point and the MCU manages feature point information in accordance with the following rules.

- An old feature point is deleted from storage if the corresponding calculation fails in successive numbers of frames.
- A new feature point is added to storage if the total number of registered points does not overflow allocated memory (200 points in our system).

We applied pin hole modeled stereo vision equations of a feature point i in the t-th frame as follows,

$$\frac{w_{t,i}}{f} \begin{bmatrix} u_{t,i} \\ v_{t,i} \\ f \end{bmatrix} = R_t^T M_i - R_t^T T_t, \quad (14)$$

where a set of $u_{t,i}, v_{t,i}$ is a position in image plane, R_t and T_t are the camera rotation and translation matrix of the earth fixed frame respectively, M_i is the feature point location matrix of the earth fixed frame, and f is the focal length of the camera. In this equation, the lens distortion of the camera internal parameter is ignored for the narrow angle of view of the CMOS camera on board.

For initialization step, depth of all feature points (the third element of M_i in eq.14) is set zero under completely planer assumption, thus its position M_i is calculated using only a frame if T_t is known. And it is iteratively updated by calculating bundle adjustment using multiple scene. After some iterations, the feature point can contribute to calculate T_t.

D. Bundle adjustment

Each feature point saved in the memory cannot have all information since it has been registered first and we use only two pieces of frame information, the first registered frame (t_i, where each t_i value is different among feature points) and the current frame (t_c) to perform bundle adjustment with

TABLE II: FPGA Utilization

Resource	Used	Available	Occupancy
Occupied Slices	$1,059$	1430	74%
Slice Registers	$1,758$	$11,400$	15%
Slice LUT	$3,338$	$5,720$	58%
BRAM 8KB	17	64	26%
DCM	2	4	50%
DSP48	0	16	0%

limited memory capacity. Under these conditions, we set the following objective function E,

$$E = \sum_{t=t_c,t_i} \sum_{i=0}^{I-1} \omega_{t,i} \{ \left(u'_{t,i} - u_{t,i}\right)^2 + \left(v'_{t,i} - v_{t,i}\right)^2 \},$$

where $\omega_{t,i}$ is the bias coefficient of weight function, the set of $u'_{t,i}, v'_{t,i}$ is the observed position of the feature point i in the t-th image plane.

The objective function E is the function with current frame of camera rotation R_{t_c} and translation T_{t_c} and each feature points location M_i. However, our system has already calculated R_{t_c} using the IMU sensor, so we used it directly for attitude estimation. Under these conditions, the MCU calculates T_{t_c} and M_i to satisfy following condition.

$$T_{t_c}, M_i = \underset{T_{t_c}, M_i}{\arg\min}\, E$$

To calculate this on the MCU, a straightforward gradient descent algorithm is applied. This small iterative calculation enables the MCU to finish all required processes in time for the update of new feature points information sent by the FPGA and thus keep a high speed estimation rate.

V. Implementation and Experiments

This section describes implementation and experiments of the visual localization system described above. The performance of our systems can also be seen in the attached video.

A. FPGA Utilization

The device utilization summary of the FPGA is shown in Table.II. The total ratio of the occupied slice is 74%, and a large of them are used for slice LUT (look-up table). And its occupancy is 58%, where large numbers of slices are used for look-up tables of intensity comparison rules of BRIEF descriptors.

We did not use any DSP (digital signal processor) units in this study, mainly because FAST and BRIEF processes do not need multiplication or division calculations.

B. Self Localization Performance

We have made the following experimental setup to measure self-localization performance. In this experiment, a localization comparison is performed under a camera depth movement. First, the CMOS camera described above and a sonar sensor are both attached to the same plane of a base object to look in the a same direction. Second, the planar basement is prepared on the ground and height measurement is performed using both methods. The MCU records both measurement results while recording the time. An additional parameter that transforms the pixel scale to object length scale is needed in our system, and we calculate this parameter by assuming that the estimated camera height and measured sonar sensor value are same at the initial step.

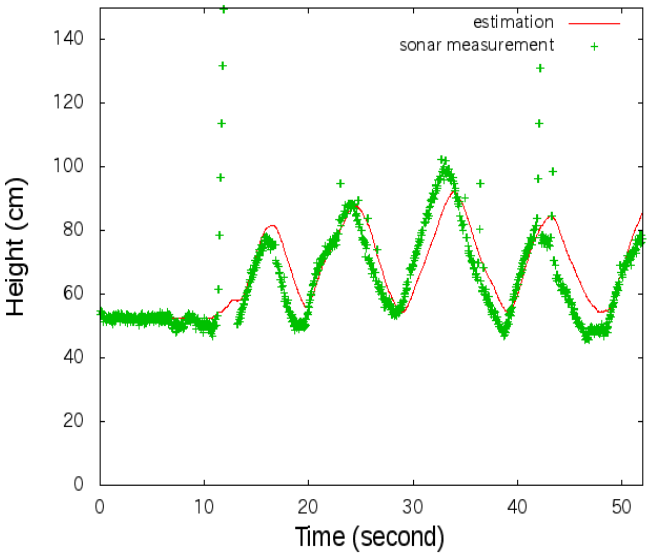

Fig. 5: Comparison to a Sonar Sensor (Dynamic Depth Movement)

The results are shown in Fig.5 and 6, where the vertical axis is the height, the horizontal axis is the time. The red line shows our localization method, and the green dots show the sonar sensor. Though the sonar sensor sometimes detects noise, other values can be regarded as ground truth measurements.

Fig.5 shows a comparison under the dynamic depth movement, and Fig.6 shows a comparison under the static depth movement ($t = 15 \sim 30s$) respectively.

Both of them clearly show accurate localization results for some periods. But travelling for a long time period accumulates small errors because no landmarks to converge them are used in our system and that is also a problem for other related methods that do not solve global SLAM problems.

VI. System Integration

A. Synthesis of a Fully Autonomous Quad-copter

We implemented the above hardware system on a palm-sized quad-copter robot whose mechanical and electronic frameworks were designed in our previous study[1] to perform fully autonomous trajectory tracking. The trajectory is programmed in the system as a group of objective points to track so that the robot can operate without any external devices. Because we do not have 3D ground truth measurement devices such as a motion capture system, the result

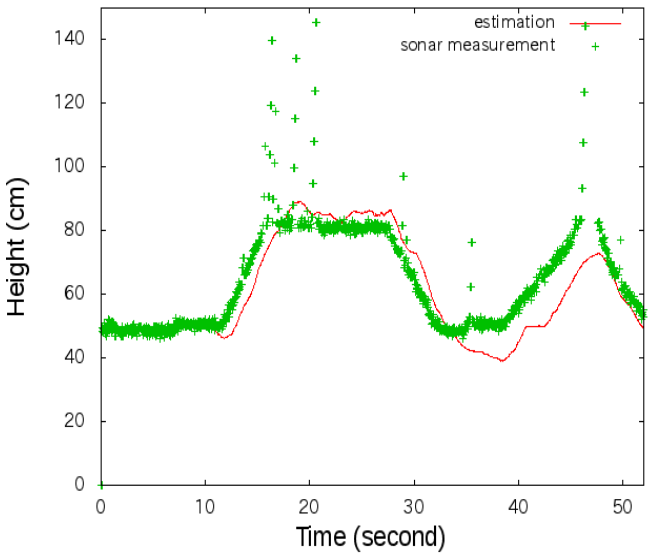

Fig. 6: Comparison to a Sonar Sensor (Static Depth Movement)

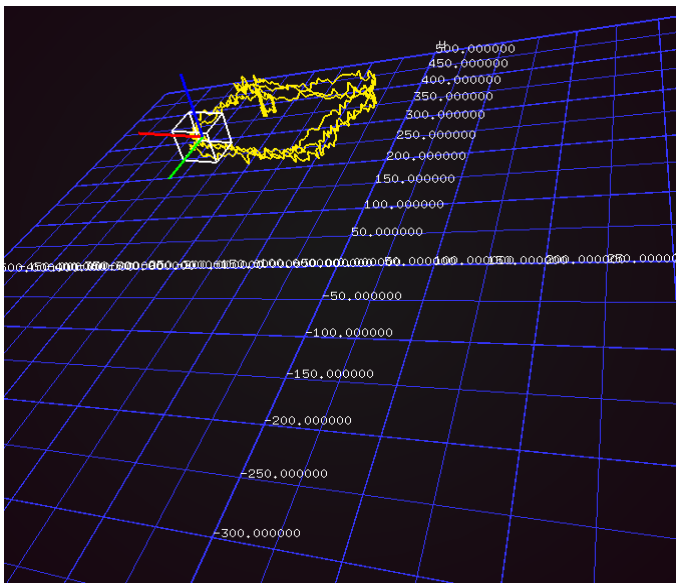

Fig. 7: On-Board and Fully Autonomous Navigation of a Rectangle Shape

shown in Fig.7 indicates only self-estimation. However, this performance can be seen in the attached video.

VII. Conclusion

We described an very cost-effective algorithm and hardware system of a 3D localization method, and the application for very compact and cost-effective hovering robots. Our proposed method can operate at low cost with very limited resource components that mainly consist of an MCU and FPGA in real time (30 frames per second) instead of applying methods requiring a large amount of calculation and memory resources with a high performance computer system. In the experiment, we compared a depth element of self-localization measurement to a ground truth measurement, and showed that an accurate estimation result can be obtained on a planer ground. We also showed the application of our method to a very compact quad-copter robot that can perform fully autonomous hovering without external assistance.

Our future plans are to improve algorithm without planer ground assumption and study the global bundle adjustment method to reduce the error in the estimated position after a long period of hovering.

References

[1] Ryo Konomura, Koichi Hori, "Designing Hardware and Software Systems Toward Very Compact and Fully Autonomous Quadrotors", IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), July 2013

[2] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," Autonomous Robots, Jan 2011

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in IEEE International Conference on Robotics and Automation, May 2011, pp. 2520–2525.

[4] Robin Ritz, Mark W. Müller, Markus Hehn, and Raffaello DAndrea, "Cooperative Quadrocopter Ball Throwing and Catching, " in IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2012.

[5] Shaojie Shen; Michael, Nathan; Kumar, V., "Autonomous multi-floor indoor navigation with a computationally constrained MAV," Robotics and Automation (ICRA), 2011 IEEE International Conference on , vol., no., pp.20,25, 9-13 May 2011

[6] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying, " in IEEE International Conference on Robotics and Automation, 2009.

[7] Blosch, M.; Weiss, S.; Scaramuzza, D.; Siegwart, R., "Vision based MAV navigation in unknown and unstructured environments," Robotics and Automation (ICRA), 2010 IEEE International Conference ,3-7 May 2010

[8] Camera-Based Navigation of a Low-Cost Quadrocopter (J. Engel, J. Sturm, D. Cremers), In Proc. of the International Conference on Intelligent Robot Systems (IROS), 2012.

[9] Inkyu Sa, Hu He, Van Huynh, Peter Corke, "Monocular Vision based Autonomous Navigation for a Cost-Effective MAV in GPS-denied Environments" IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), July 2013

[10] M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In-and Outdoor Environments, " in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2011.

[11] Stephan Weiss, Markus W. Achtelik, Simon Lynen, Margarita Chli, Roland Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," Robotics and Automation (ICRA), 2012 IEEE International Conference pp.957, May 2012

[12] B. herisse, F. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow, " in International Conference on Intelligent Robots and Systems, Sept. 2008, pp. 801806.

[13] Hyon Lim; Hyeonbeom Lee; Kim, H.J., "Onboard flight control of a micro quadrotor using single strapdown optical flow sensor," Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on , vol., no., pp.495,500, 7-12 Oct. 2012

[14] Georg Klein and David Murray, "Parallel Tracking and Mapping for Small AR Workspaces", International Symposium on Mixed and Augmented Reality (ISMAR'07, Nara)

[15] E. Rosten, R. Porter, and T. Drummond. "Faster and better: A machine learning approach to corner detection." IEEE Trans. Pattern Analysis and Machine Intelligence, 32:105119, 2010.

[16] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "Brief: Binary robust independent elementary features." In European Conference on Computer Vision, 2010.

[17] Madgwick, S.O.H.; Harrison, A. J L; Vaidyanathan, R., "Estimation of IMU and MARG orientation using a gradient descent algorithm," Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on , vol., no., pp.1,7, June 29 2011-July 1 2011