MVA2011 IAPR Conference on Machine Vision Applications, June 13-15, 2011, Nara, JAPAN

# Quadrotor Helicopter Flight Control Using Hough Transform and Depth Map from a Microsoft Kinect Sensor

John Stowers
University of Canterbury
john.stowers@ieee.org

Michael Hayes
University of Canterbury
michael.hayes@canterbury.ac.nz

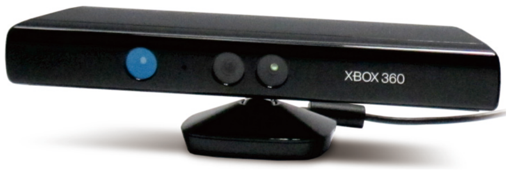
Andrew Bainbridge-Smith
University of Canterbury
andrew.bainbridge-smith@canterbury.ac.nz

## Abstract

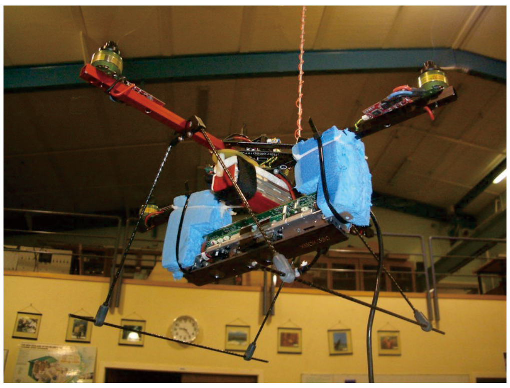
*Reliable depth estimation is important to many autonomous robotic systems and visual control algorithms. The Microsoft Kinect is a new, low cost game controller peripheral that calculates a depth map of the environment with good accuracy and high rate. In this paper we use the calibrated output of the depth sensor to obtain an accurate absolute depth map. Subsequently by using a Randomized Hough Transform to detect the ground plane in this depth map we are able to autonomously hover a quadrotor helicopter.*

## 1 Introduction

The Microsoft Kinect (Figure 1a) is a low cost computer vision peripheral, released November 2010, for use with the Xbox 360 game system as a game controller. The device can be modified to obtain, simultaneously at 30 Hz, a $640 \times 480$ pixel monochrome intensity coded depth map and a $640 \times 480$ RGB video stream.



(a)



(b)

Figure 1: (a) An unmodified Kinect. From left to right the sensors are; the IR projector, RGB camera, and monochrome (depth) camera. (b) The modified Kinect mounted on the quadrotor helicopter.

Depth maps are employed in robotic control systems for autonomous navigation [1], map building [2, 3] and obstacle avoidance [4]. Due to this ubiquity, and as little public information is available from the manufacturer, this paper quantifies the accuracy of the depth map provided by the Kinect sensor and subsequently uses this data to control the altitude and hover an autonomous quadrotor helicopter. In performing this work we demonstrate the Kinect sensor and its suitability for use in dynamic robotic environments.

This paper is structured as follows. Section 1 introduces the Kinect sensor hardware and the use of depth maps in research. Section 2 explains the calibration procedure and calibration results. Section 3 introduces the problem of identifying planes in range data and describes the randomized Hough transform. Section 4 introduces the quadrotor experimental platform and control system. The paper concludes with Section 5, a discussion of experimental flight results using the sensor and further work.

### 1.1 Computation of Depth Maps

The computation of depth maps can be grouped into passive or active methods [5]. Passive depth sensing tries to infer depth from multiple cameras or images, for example, through stereo correspondence algorithms or optical flow. Active methods usually employ additional physical sensors such as lasers, structured lighting, or infra–red illumination cast on the scene.

The Kinect uses a form of structured light.[1] The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor. Little information is available on the structured light method used, or the accuracy of the depth map obtained from the Kinect.

### 1.2 Kinect Hardware Details

The Kinect sensor connects to the PC/XBOX using a modified USB cable.[2] However the USB interface remains unchanged, and subsequent to the Kinect release, the protocol[3] was decoded and software to access the Kinect was created.

The Kinect features two cameras, a Micron MT9M112 $640 \times 480$ pixel RGB camera and a

[1]developed and patented by PrimeSense; `http://www.primesense.com/?p=535`, often referred to as ‘a proprietary Light Coding$^{TM}$technology’.

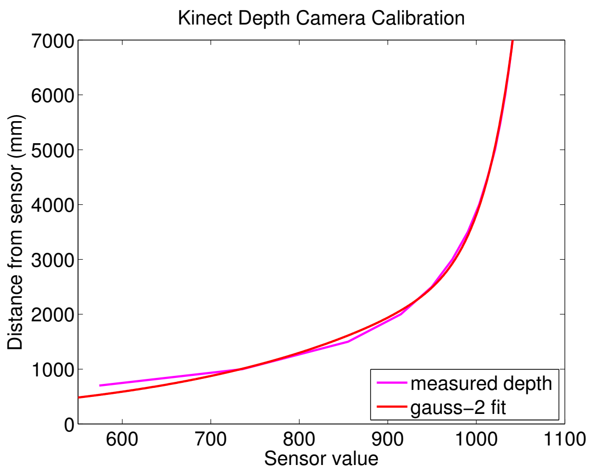
[2]required to provide additional current

[3]gratefully started by the OpenKinect project; `https://github.com/OpenKinect`

1.3 megapixel monochrome Micron MT9M001 camera fitted with an IR pass filter. Accompanying the monochrome IR camera is a laser diode for illuminating the scene. The depth map has 11-bit resolution and the video hardware 8-bit resolution. Both cameras deliver $640 \times 480$ pixel images at 30 Hz and have an angular field of view of 58° horizontally, 45° vertically and 70° diagonally. The spatial, $(x, y)$, resolution (at 2 m from the sensor) is 3 mm while the depth, $z$, resolution at the same distance is 10 mm.[4]

## 2 Calibration of the Kinect Depth Camera

The depth camera returns an 11-bit number which needs further processing in order to extract the true depth. If the Kinect is unable to estimate the depth in certain regions in the image, those pixels are filled with the special value 2047. To calibrate the depth sensor a number of reference images, which featured a prominent plane at known distance, were captured. The process was repeated multiple times in different ambient and artificial lighting conditions. The results of this calibration procedure are shown in Figure 2a.



(a)

Figure 2: Kinect depth camera calibration results and line of best fit.

A second order Gaussian model was found to be an appropriate fit ($r^2 = 0.9989$) for the data over the tested range of 0.4 – 7.0 m.[5] Let $f(x)$ be the true depth from the image sensor, and $x$ the raw range value, then

$$\begin{aligned} f(x) &= a_1 \exp(-((x-b_1)/c_1)^2) \\ &+ a_2 \exp(-((x-b_2)/c_2)^2). \end{aligned} \tag{1}$$

Control of the quadrotor occurs in the world coordinate frame, yet the image data returned from the Kinect is specified in the image coordinate frame (Figure 3). To convert from one frame to another the standard projection using homogeneous coordinates is used;

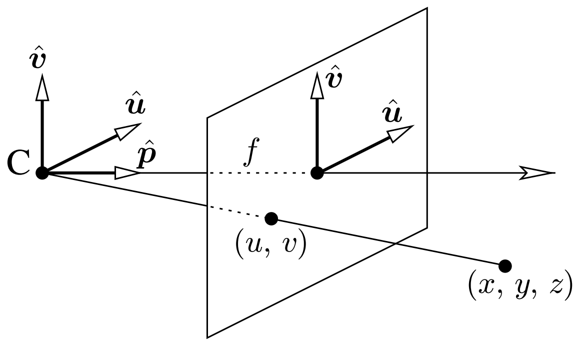


Figure 3: Pinhole camera geometry. $C$ is the camera centre, $P$ the principle axis, and $f$ the focal length. $(u, v)$ is a pixel in the image plane, $(x, y, z)$ a point in the world, and $d(u, v)$ the depth reading from the Kinect.

$$\begin{bmatrix} u \\ v \\ d \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2}$$

with $u, v$ the pixel indices, $d$ the depth returned from the Kinect and $(c_x, c_y)$ the co-ordinates of the principal point in the image frame. Let dist($d$) return the calibrated distance in meters and rearrange (2) for $(x, y, z)$ to give;

$$z = \text{dist}(d) \tag{3}$$
$$y = \frac{(u - c_y)z}{f_y} \tag{4}$$
$$x = \frac{(u - c_x)z}{f_x}. \tag{5}$$

Table 1 includes the depth calibration and intrinsic camera calibration parameters.

| | |
|---|---|
| $a_1 = 3.169 \times 10^4$ | $a_2 = 6.334 \times 10^{18}$ |
| $b_1 = 1338.0$ | $b_2 = 2.035 \times 10^4$ |
| $c_1 = 140.4$ | $c_2 = 3154.0$ |
| $f_x = 594.21$ | $f_y = 591.04$ |
| $c_x = 339.5$ | $c_y = 242.7$ |

Table 1: Kinect calibration parameters; Gaussian fit coefficients for depth $(a_1, a_2, b_1, b_2, c_1, c_2)$ and camera intrinsic parameters $(f_x, f_y, c_x, c_y)$.

## 3 Identifying Planes

Plane detection is the problem of estimating a planar surface from a cloud of 3D points, $\mathcal{P} = (\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n})$. In the case of autonomous hovering presented in this paper, it is the ground plane that is of interest. We need to detect the ground plane in the calibrated point cloud and thus recover our altitude.

Consider a plane, Figure 4. To avoid problems with infinite slopes we use the normal form of the plane; given by a point $\mathbf{p} = (p_x, p_y, p_z)$ on the plane, a normal vector $\mathbf{n} = (n_x, n_y, n_z)$ perpendicular to the plane, and the distance $\rho$ to the origin. The equation of the plane is

$$\rho = \mathbf{p} \cdot \mathbf{n} \tag{6}$$
$$\rho = p_x n_x + p_y n_y + p_z n_z. \tag{7}$$

[4]provided by the PrimeSense reference design for the PS1080 chipset used in the Kinect. `http://www.primesense.com/?p=514`

[5]the official PrimeSense documentation states the operational range of the sensor is only 0.8 m – 3.5 m

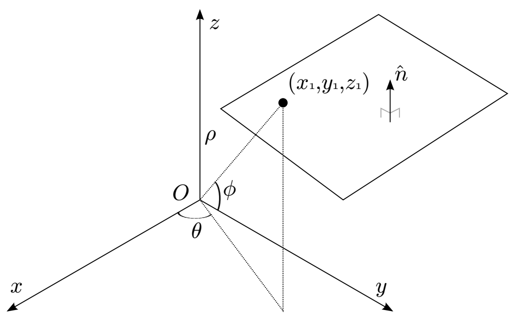


Figure 4: Presentation of a plane and its normal $\mathbf{\hat{n}}$ in 3D space. $(x_1, y_1, z_1)$ is a point on the plane.

Consider Figure 4. Taking into account the angles between the normal vector and the co-ordinate axis, (7) can be written

$$p_x \cos\theta \sin\phi + p_y \sin\theta \sin\phi + p_z \cos\phi = \rho. \quad (8)$$

The angles $\theta$, $\phi$, and distance $\rho$ parametrise the plane. The plane estimation problem can thus be posed as the need to estimate $\theta$, $\phi$, and $\rho$ from clouds of measured points. There are a number of techniques to solve this problem; such as those based on the Hough transform, explained in the following section.

## 3.1 The Hough Transform

The Hough transform [6] is a method for detecting parametrised objects in data; such as lines or ellipses in the 2D case and planes in the 3D case. The principle of this technique is the mapping of a set of points initially defined in Euclidean space into another parameter space; often called the Hough space. In doing so, certain geometric primitives can be detected with improved computational efficiently.

For example, consider the normal form of a line;

$$\rho = x\cos\theta + y\sin\theta, \quad (9)$$

where $\theta$ and $\rho$ are the parameters of the normal passing through the origin (Figure 5a).

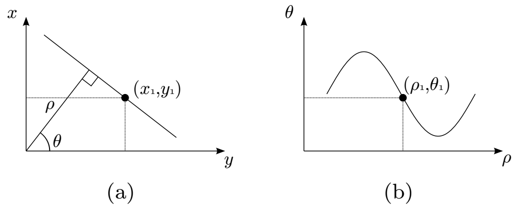


Figure 5: (a) presentation of line and its normal in 2D space. (b) the same point now in the parameter space, represented using the normal form.

The parameter space in this case is $(\theta, \rho)$ and one point $\mathbf{p_1} = (x_1, y_1)$ in 2D space represents a sinusoid in parameter space (Figure 5b). Detection of multiple sinusoid curves passing through the same point in parameter space allows one to detect straight lines in 2D space.

The same concept can be applied in the 3D case [7, 8] to detect planes in range data. Consider the plane in (8) whose parameter space is $(\theta, \phi, \rho)$. One plane in 3D space represents a sinusoidal surface in the parameter space. Detection of multiple similar sinusoidal surfaces allows one to detect planes in 3D space.

For detection the parameter space is discretized into $I$, $J$, $K$ values for $\theta'$, $\phi'$, $\rho'$. A datastructure, usually called an accumulator $(A)$ stores a score for each of these cells. In the standard Hough transform (SHT) each point votes for all sets of parameters $(\theta', \phi', \rho')$ on which it may lie. After all points have been processed, the cells with the highest values represent the most prominent planes; those that cover the most points of the cloud.

The SHT has many limitations for real-time use. It has high computational complexity; for an $M \times N$ image it is of the order $O(MNIJK)$ [9]. The determination of peaks in the accumulator is difficult; discretization means planes may occupy many adjacent cells, so a sliding window type search must be used to find the most prominent regions. It also has high memory requirements since a 3D accumulator has $P \times Q \times R$ cells for storage.

## 3.2 The Randomized Hough Transform

Randomized Hough Transform (RHT) [10] based plane detection begins with the premise that a single plane can be determined uniquely with three points from the range data. These three points in 3D space are mapped into one point in the parameter space corresponding to the plane spanned by the three points.

The parameter space $(\theta, \phi, \rho)$ is discretized into $I$, $J$, $K$ values. In each iteration, three points $\mathbf{p_1}$, $\mathbf{p_2}$, and $\mathbf{p_3}$ are randomly selected from $\mathcal{P}$. The plane spanned by these points is calculated with the cross product

$$\mathbf{n} = (\mathbf{p_3} - \mathbf{p_2}) \times (\mathbf{p_1} - \mathbf{p_2}). \quad (10)$$

This gives the normal vector of the plane spanning the three points. The unit vector to the plane is

$$\mathbf{\hat{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|}, \quad (11)$$

where $\mathbf{\hat{n}} = (n_x, n_y, n_z)$ and using (7) we compute $\rho$ as

$$\rho = \mathbf{\hat{n}} \cdot \mathbf{p_1}. \quad (12)$$

Since $\mathbf{\hat{n}}$ is a unit normal it can be represented by only two parameters in spherical polar form

$$\begin{aligned} n_x &= \cos\theta\sin\phi \\ n_y &= \sin\theta\sin\phi \\ n_z &= \cos\phi, \end{aligned}$$

and by rearranging we can compute the remaining plane parameters

$$\phi = \cos^{-1} n_z \quad (13)$$

$$\theta = \sin^{-1} \frac{n_y}{\sin\phi}. \quad (14)$$

The corresponding cell in the accumulator $A(\theta, \phi, \rho)$ is incremented. If a threshold $T_A$ is exceeded by the score in the cell then a plane is detected. Otherwise, the algorithm continues until all the points have been processed or a maximum number of iterations $T_I$ is reached.

The computational complexity of finding the biggest plane with area $m$ is approximately $O(\min(m^3 T_A, T_I))$, which is independent of the size of image and quantification steps [9, 11]. Compared to the SHT, the RHT can detect planes more efficiently.

### 3.3 Identifying the Ground Plane

The biggest plane has the largest probability of being detected and because of the orientation of the Kinect camera (looking predominately at the ground), the biggest plane should be the ground plane.

Suppose $\frac{1}{m^{th}}$ of the points in $\mathcal{P}$ lie on the biggest planar surface. The probability of the three randomly selected points simultaneously belonging to the biggest plane is thus $\frac{1}{m^3}$.

The algorithm is implemented as described in Section 3.2, but with the following extensions:

1. Unlike the canonical implementation [10], this implementation does not back–calculate all $\mathcal{P}$ points that correspond to $A(\theta, \phi, \rho)$ once $T_A$ is reached. Because we only want to detect the largest plane (the ground plane), the algorithm terminates once $T_A$ is met.

2. A distance criteria is introduced as recommended by Borrmann *et al.* [12]; $\text{dist}_{\text{max}}(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}) \leq \text{dist}_{\text{max}}$, where dist is the Euclidian point–to–point–to–point distance of all three points and $\text{dist}_{\text{max}} = 7\,\text{m}$ (a first order approximation is the maximum range of the Kenect sensor, however this can be reduced based on assumptions about the environment, such as maximum altitude). Only points randomly chosen from $\mathcal{P}$ that fulfil this criterion are evaluated thus minimizing computation time, as only points that are reasonably close are considered as candidates for the ground plane.

3. In addition to the distance criterion test applied to the randomly selected candidate points, we introduced a second constraint. While the first point $\mathbf{p_1}$ is randomly selected from $\mathcal{P}$, points $\mathbf{p_2}$ and $\mathbf{p_3}$ are randomly selected from a smaller sub-region of the image surrounding $\mathbf{p_1}$. The reduces computation time by helping to prevent selecting candidate points that would fail the distance criteria test anyway.

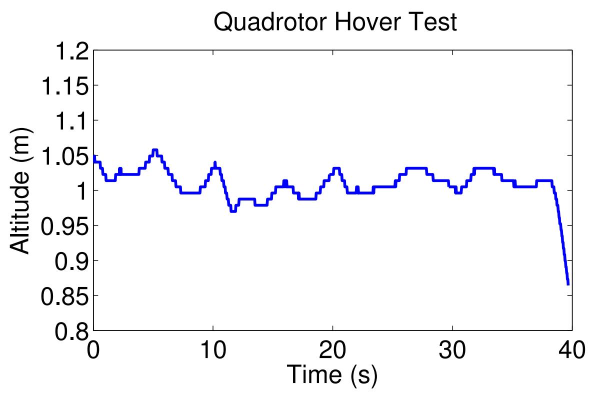
Figure 6 shows the algorithm applied to a single image from a sequence captured during flight.

## 4 Quadrotor Flight Control

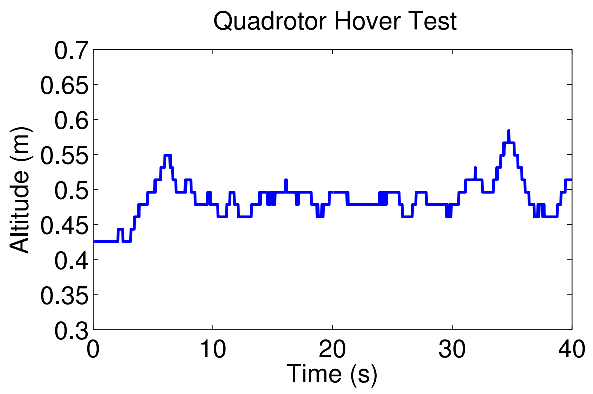
### 4.1 Experimental Hardware

The quadrotor (Figure 1b) is of custom design and construction [13]. It features a real time embedded attitude controller and onboard inertial measurement unit (IMU) which contains a 3-axis accelerometer, 3x single-axis gyroscopes, and a 3-axis magnetometer.

The Kinect sensor is mounted under the craft, pointing towards the ground. The visual controller runs on a standard laptop PC. The Kinect is connected to the PC via USB which means the quadrotor is limited to operation within cable range of the laptop.



(a)



(b)

Figure 7: Hovering performance; (a) Setpoint = 1.0 m. (b) Setpoint = 0.5 m.

### 4.2 Flight Control System

We assume that the ground is flat and the output from the Kinect is correct, thus we let $\rho = Pv$.

A proportional-integral (PI) controller was implemented to control the quadrotor altitude. The commanded output, $c$, from the controller is given by;

$$c = K_P \Delta + K_I \int \Delta \, dt, \tag{15}$$

where $K_p = 5$ and $K_I = 1$ are the proportional and integral control gains determined experimentally, $\Delta$ is the difference between the commanded ($Sp$) and measured ($Pv$) altitude; $\Delta = Sp - Pv$. The control system is discrete, a 4th–order Runge-Kutta (RK4) integrator is used and $dt$, the frequency of update of the control system, is $\frac{1}{20\,\text{Hz}} = 50\,\text{ms}$.

## 5 Results

The RHT algorithm was developed using MATLAB and then ported to c++ for the control experiments. The visual flight controller was given complete authority to command the quadrotor thrust, and hence its altitude. Figure 7 shows the performance of the control system. A video[6] is also available.

The quadrotor was commanded to hover at fixed altitudes above the laboratory floor. The oscillation in

[6]`http://www.waspuav.org/resources/mva2011/kinect-hover-video`

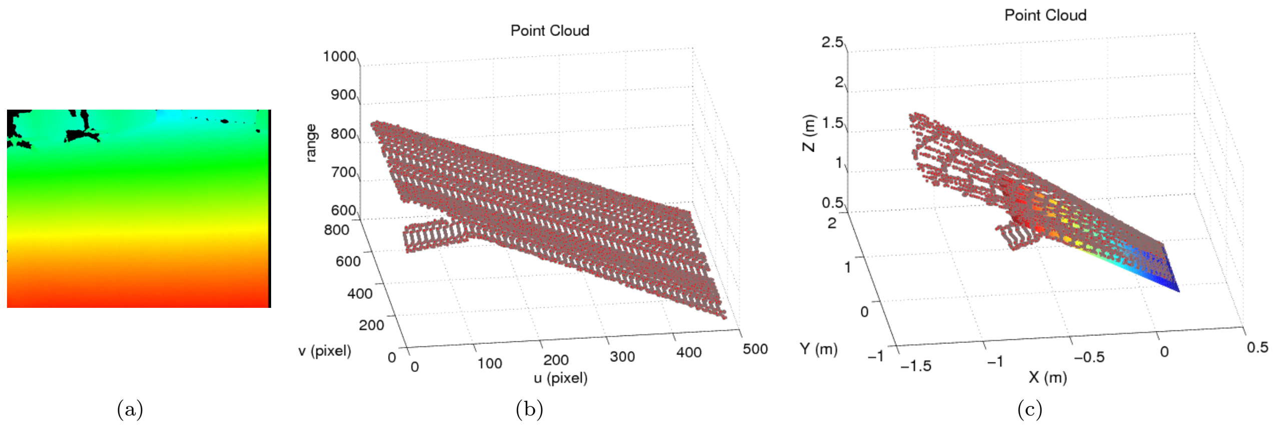


Figure 6: Processing stages in detecting the ground plane. (a) False color image of obliquely mounted camera observing the ground. (b) Range data point cloud in the camera frame. (c) Point cloud transformed into world coordinate frame with the detected ground plane overlaid.

altitude shows that while the PID controller is not optimal for this task, it allowed the quadrotor to regulate attitude for the duration of the tests (40 s).

The performance of the RHT was very good. In order to detect the ground plane, the mean number of planes examined (incremented in the accumulator) was $35 \pm 14$ ($1\sigma$). With a suitable value for $\mathrm{dist}_{\max}$, execution time of the algorithm was $\approx 10$ ms and it was found that on average, $\ll 2\%$ of the image points were tested.

## 6 Conclusion

We demonstrated successful control of quadrotor altitude using the Kinect depth map and RHT. We showed that the RHT is very efficient at detecting the ground plane. Finally, we also showed that the Kinect is suitable for operation in dynamic environments and that it works effectively, with accurate calibration, out to a range of 7 m.

## References

[1] D. Murray and J. J. Little, "Using real-time stereo vision for mobile robot navigation," *Autonomous Robots*, vol. 8, no. 2, p. 161, 2000.

[2] D. Wooden, "A guide to vision-based map building," *Robotics Automation Magazine, IEEE*, vol. 13, no. 2, pp. 94–98, june 2006.

[3] R. Sim and J. Little, "Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters," *Image and Vision Computing*, vol. 27, no. 1-2, p. 167, 2009.

[4] M. Kumano, A. Ohya, and S. Yuta, "Obstacle Avoidance of Autonomous Mobile Robot using Stereo Vision Sensor," in *Proceedings of the 2nd International Symposium on Robotics and Automation*, 2000, pp. 497–502.

[5] S.-Y. Kim, E.-K. Lee, and Y.-S. Ho, "Generation of ROI Enhanced Depth Maps Using Stereoscopic Cameras and a Depth Camera," *Broadcasting, IEEE Transactions on*, vol. 54, no. 4, pp. 732–740, dec. 2008.

[6] P. Hough, "Method and Means for Recognizing Complex Patterns," US Patent 3069654, 1962.

[7] J. Overby, L. Bodum, E. Kjems, and P. M. Ilsøe, "Automatic 3D building reconstruction from airborne laser scanning and cadastral data using Hough transform," *Int. Arch. of Photogrammetry and Remote Sensing*, vol. 35, part B3, 2004.

[8] G. Vosselman, E. Dijkman, K. W. B. Reconstruction, L. Altimetry, and H. Transform, "3D building model reconstruction from point clouds and ground plans," *Int. Arch. of Photogrammetry and Remote Sensing*, pp. 37–43, 2001.

[9] L. Xu and E. Oja, "Randomized Hough transform (RHT): basic mechanisms, algorithms, and computational complexities," *CVGIP: Image Underst.*, vol. 57, no. 2, pp. 131–154, March 1993.

[10] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized Hough transform (RHT)," *Pattern Recogn. Lett.*, vol. 11, no. 5, pp. 331–338, May 1990.

[11] Y. Ding, X. Ping, M. Hu, and D. Wang, "Range image segmentation based on randomized Hough transform," *Pattern Recognition Letters*, vol. 26, no. 13, pp. 2033–2041, 2005.

[12] K. L. Dorit Borrmann, Jan Elseberg and A. Nüchter, "A Data Structure for the 3D Hough Transform for Plane Detection," in *Proceedings of the 7th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '10)*, Lecce, Italy, 2010.

[13] J. Stowers, M. Hayes, and A. Bainbridge-Smith, "Quadrotor Helicopters for Visual Flight Control," in *Proceedings of Electronics New Zealand Conference 2010*, Hamilton, New Zealand, 2010, pp. 21–26.