# Real-Time Robust Tracking with Commodity RGBD Camera

Abdenour Amamra and Nabil Aouf
Department of Informatics and Systems Engineering, Cranfield University, UK
{a.amamra, n.aouf}@cranfield.ac.uk

***Abstract*—Commodity RGBD cameras such as Kinect sensor have recently proven a large success in many indoor robotics and computer vision applications. Nevertheless, tracking and motion estimation algorithms cannot rely on Kinect raw outputs because of their low accuracy. These consumer cameras can only produce precise depth measures within a close range. However, they do suffer from potential noises when the target is further away from permitted. This paper proposes an innovative adaptation of Kalman filtering scheme to improve the accuracy of Kinect as a real-time tracking device. We present a detailed proof of Kalman filter adaptation on Kinect data, and we demonstrate the robustness of our approach on a real dataset.**

***Keywords-Kinect; Kalman filter; real-time tracking; obstacle avoidance;***

## I. INTRODUCTION

To avoid colliding with potential obstacles, and to get a decent visual tracking, motion estimation and obstacle avoidance algorithms should be able to get the correct data at the right time. That is, both accuracy and response time are to be considered as a first priority when designing tracking and navigation systems [1]. Until now, most of the available solutions in the literature are based on classical intensity cameras and rangefinders. Whether range or vision-based, each of the two approaches has some advantages and drawbacks. The former is based upon two entities, radar or laser waves. While cheap and ubiquitous, ultrasonic rangers, are plagued with fundamental difficulties that make them unattractive for serious localisation. On the other hand, laser rangefnders use light instead of sound to determine the distance from time-of-flight. These sensors are often bundled with precision optics and mechanisms that allow them to infer 3D point clouds. Their precision and reliability make them ideal for industrial and research applications, but their cost is prohibitive for low budgets [2]. Vision-based approaches can provide disparity information after extracting features and applying some algorithms such as stereo, structure-from-motion, visual SLAM, and visual odometry [1]. These algorithms are computationally expensive though. However, the increasing capabilities of the new processing units, and the assumptions we could make about the target applications, can significantly improve the response time [3]. Moreover, these sensors have a relatively low cost compared to laser scanners, and decent state-of-art solutions are already available and scalable to most configurations [3].

Our choice of Kinect sensor[1] is motivated by the fact that it combines both range and intensity in real-time (30fps), at a very compelling price. We developed a filtering stage which improves the sensor's raw measurements, without losing its real-time asset. Consequently, the raw output is optimally filtered, and accuracy in measurements is noticed as will show the provided experimental results. The application of the filter is not yet straightforward. Thus, a thorough analysis of the sensor's data is presented with its corresponding mathematical formulation.

## II. RELATED WORK

Fewer works were conducted on tracking and navigation with Kinect. Bachrach et al. enabled an Unmanned Aerial Vehicle (UAV) to explore indoor environments with Kinect, where GPS data are either weak or not available [4]. Endres et al. created a system performing SLAM and 3D reconstruction [5]. The authors reported an RMSE of 9.7cm in a typical office environment. Ruhnke et al. optimised Kinect's pose and the positions of the measured surface points in order to create more accurate 3D reconstructions [6]. On the Other hand, intensity based tracking is better studied and good results were already achieved with systems using multiple high resolution cameras. Valenti et al. worked on the development of RAVEN system which aims to estimate the position of a UAV in indoor environment [7]. Yoshihata et al. used two stationary and upward-looking cameras placed on the ground to track four black balls attached to a flying helicopter [8].
Our contribution falls in the field of real-time, object tracking (fixed viewpoint) and vision-based navigation (moving viewpoint) with one RGBD sensor (Microsoft Kinect).

## III. KINECT CAMERA

### A. Technical specifications of the sensor

Kinect sensor is an RGBD camera which has the ability to capture both the depth map of the scene, and its RGB colour image in real-time. The sensor includes (Figure 1):

- An IR-projector: projects an IR pattern on the scene.
- An IR-camera: captures the reflected light of the projected pattern.
- An RGB camera: works as an ordinary colour camera.

[1]Microsoft Kinect : http://www.xbox.com/en-GB/Kinect, 2012

978-1-4799-0652-9/13 $31.00 © 2013 IEEE
DOI 10.1109/SMC.2013.411

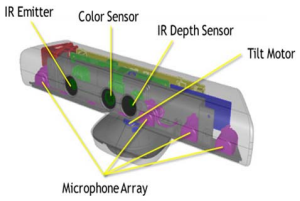


Figure 1. Kinect sensor

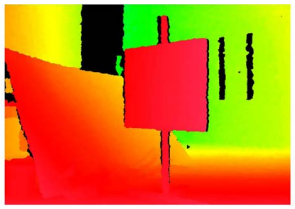

Figure 2. Kinect Raw data

Kinect captures video frames at a rate of 30 frames per second, and computes depth with a triangulation process Figure 2. After a calibration procedure, both RGB image and IR depth data can be fused to form a coloured 3D point cloud of about 300,000 points in every frame[2]. Like any camera, Kinect sensor cannot cover the entire continuous resolution of the physical scene [9]. It actually projects the captured depth data on a set of discrete parallel planes. Hence, some data which should be positioned between the planes (according to its real world continuous $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ coordinates) is either lost or shifted to a neighbouring discrete range level. The accuracy of the sensor is largely affected by this behaviour. More importantly, if we use Kinect in tracking applications, this drawback will adversely affect the measurements and results in significant damages to the tracked entities. The innovative point in our research work is that, inspired by this 3D structure of the sensor's output, and using a well-established filtering scheme (Kalman Filter [10]), we were able to improve the capabilities of Kinect as a tracking device, and to make the measurements steadier and more reliable over time. However, the adaptation of the sensor's output to Kalman filter is not possible before the satisfaction of some conditions. The noise we want to reduce should be white and Gaussian. The entity to optimise (depth measurement) should evolve linearly over time. In the following section, we present a thorough analysis followed by the proof that all the requirements to apply the filter are fulfilled with Kinect data.

### *B. Z-resolution*

To study the nature of depth resolution, we pointed the sensor at a parallel direction along with a large flat wall Figure 3. This setup allows us to capture a cloud of points from the whole operating range of Kinect, and to characterise the interesting properties for the filter.

As shown in Figure 4, the depth resolution is inversely proportional to the distance from the sensor. In addition, the points within the capture (taken from one frame) are distributed on independent clusters which we will call "**Z-Levels**". Accordingly, we formulate Kinect's data as a finite

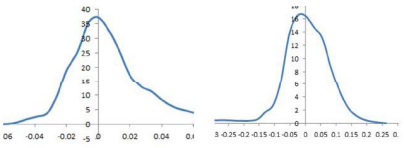

Figure 6. Kinect noise distribution, at 1.512m and 3.406m respectively

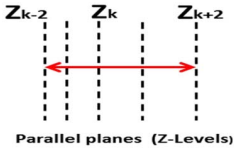


Figure 7. Deviation range ***5 Z-Levels***

set of points distributed on parallel planes (**Z-Levels**), where every plane constitutes a partition of the whole capture.

The mathematical definition is as follows:
$\boldsymbol{I_k}$ : Set of indices ranking the parallel planes.
$\boldsymbol{I_i}$ : Set of indices indexing the points lying in the planes.
$\boldsymbol{C}$ : Set of the whole point cloud data Figure 5,
$\boldsymbol{Z_k}, \boldsymbol{k} \in \boldsymbol{I_k}$ : Plane in $\boldsymbol{C}$ Figure 5,
$\boldsymbol{P_i}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i}), \boldsymbol{i} \in \boldsymbol{I_i}$ : Point in RGBD space, lying in a given Z-Level; $\boldsymbol{Z_k} = \mathbf{z_i}, \boldsymbol{k} \in \boldsymbol{I_k}$, Figure. 5,
Every point cloud $\boldsymbol{C}$ satisfies the properties:

$$C = \bigcup_k Z_k, k \in I_k \quad (1)$$

$$\forall Z_{k1}, Z_{k2} \in C, k_1, k_2 \in I_k, k_1 \neq k_2 ; Z_{k1} \cap Z_{k2} = \emptyset \quad (2)$$

$$\forall p_i \in C, i \in I_i, k \in I_k ; \exists! Z_k, p_i \in Z_k \quad (3)$$

$$\forall Z_k \in C, k \in I_k ; Z_k \perp Z_{axis} \quad (4)$$

### *C. Depth Noise statistics*

Kinect as an electronic device has a hardware related noise. This noise can come from reference template accuracy, calibration process, the lighting conditions and the objects' surface properties [9]. Errors in the projected data increase with increasing inter Z-Levels distances, because of depth resolution decrease Figure 4. A study, we conducted to discover the nature of noise affecting the depth measurement, showed that it has a Gaussian distribution, with varying standard deviation depending on the range from the sensor. Figure 6 shows some samples we took at 1.512m and 3.406m from the sensor. Based on the graphs, we can extract the corresponding standard deviations which were 0.02m and 0.075m, respectively.
When we re-projected the sampled points to their original depth map, we found that the standard deviations $\boldsymbol{\sigma_k}$:

$$\sigma_k = (Z_{k+i} - Z_{k-i}) / 2, \forall k \in I_k, i \in \mathbb{N} \quad (5)$$

$\sigma_k$ is the average distance between the two extremities of the ***2i+1*** successive Z-Levels and the central level $Z_k$ to which belongs the sampled point Figure 7. As a result, at every Z-Level $Z_k$, Kinect noise remains Gaussian and $\sigma_k$ is its

Figure 3. Setup's RGB output

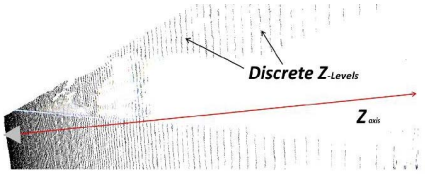


Figure 4. Setup's depth output

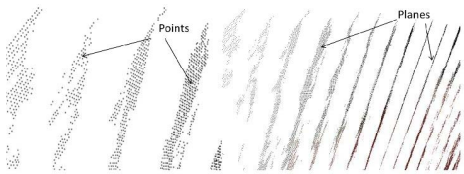


Figure 5. Point cloud components

[2]Willow Garage. Turtlebot. http://www.willowgarage.com/turtlebot, 2011

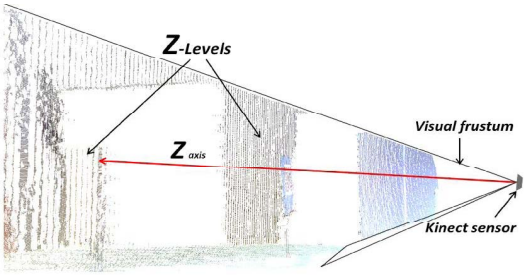

Figure 8. Distribution of Z-levels

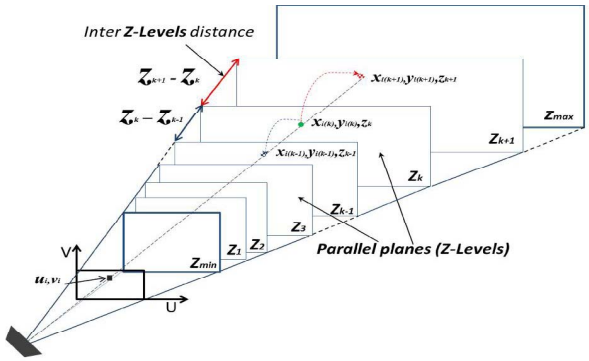

Figure 9. Inter ***Z-Levels*** linear displacement of a given point towards the perspective frustum lines

standard deviation (5).

This property allowed us to prove the Gaussian nature of noise affecting the depth data, and to consequently satisfy the first condition to apply Kalman filter. Moreover, we can precisely attribute a standard deviation $\sigma_k$ to every Z-Level which will serve to compute optimal depth measurements for all the points in the cloud.

### *D. States of an IR pixel*

When we point the sensor against a static scene, and we observe the depth map over time Figure 8 (both the sensor and the scene remain steady during the whole capture), we notice fluctuations in almost 90% of all map's elements. The change is limited within a finite set of values close to each other. If we closely observe the rendered points, we notice that they tend to disappear from one Z-level and reappear in a neighbouring one as illustrated in Figure 9. In addition, for every capture of any scene, there is a finite set of repetitive depth values. In other words, we can predict the possible discrete depth values that we may encounter in the output data. As explained above, Kinect sensor works in a discrete set of depth elements that we called "**Z-Levels**". Every level constitutes a partition of the whole set of points within a frame (1), (2), (3), has the properties of being completely independent from the neighbouring levels and orthogonal to $\boldsymbol{Z_{axis}}$ (4). As a result, we cannot find a point out of these parallel planes. This is what really appears in all the captured data if we rotate the scene over $\boldsymbol{X_{axis}}$ or $\boldsymbol{Y_{axis}}$ (the points lie in planes parallel to $\boldsymbol{XY}$). The importance of such information for Kinect based applications is that we can study the relationship between depth measurements taken over different frames. That is, if we find that two successive depth measures are related with a linear function, we should have fulfilled the second condition required to use Kalman filter.

When the sensor is stationary, the depth map keeps changing because of points jumping from one Z-Level to another Figure 9, not necessarily adjacent, but within a limited radius (5).

When the points change their depth level, their 2D $(\mathbf{u_i}, \mathbf{v_i})$ image coordinates on the screen remain the same, but their world coordinates $(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i})$ change. This is true because every point $\boldsymbol{P_i}$ $(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i})$ in the 3D world lies on a line incident from the centre of the camera passing through the pixel $(\mathbf{u_i}, \mathbf{v_i})$ on the screen toward the scene Figure 9, following the direction of the perspective frustum [11]. Using this information, we can get the relationship between two successive measured 3D coordinates.

From the intrinsic parameter of the camera $(\boldsymbol{f_x}, \boldsymbol{f_y}; \boldsymbol{c_x}, \boldsymbol{c_y})$. We have these equations ((+) means the new coordinates in the following frame):

From camera calibration model we have:

$$\begin{cases} u_i = (f_x/z_i)x_i + c_x \\ v_i = (f_y/z_i)y_i + c_y \end{cases} \quad z_i \neq 0 \tag{6}$$

As the pixel coordinates in two successive frames remain the same, from (6) we get with $z_i \neq 0, z_i^+ \neq 0$:

$$\begin{cases} u_i^+ = u_i \xrightarrow{yields} (f_x/z_i^+)\, x_i^+ + c_x = (f_x/z_i)x_i + c_x \\ v_i^+ = v_i \xrightarrow{yields} (f_y/z_i^+)\, y_i^+ + c_y = (f_y/z_i)y_i + c_y \end{cases} \tag{7}$$

And finally, after simplification of (7):

$$\begin{cases} x_i^+ = \left(\dfrac{z_i^+}{z_i}\right) x_i \\ y_i^+ = \left(\dfrac{z_i^+}{z_i}\right) y_i \end{cases} \tag{8}$$

Equation (8) proves the **linearity** between points projected at the same pixel on the screen over different frames. Adding this to the Gaussian nature of noise, we can safely use Kalman model as a real-time filtering method.

## IV. KALMAN FILTER ADAPTATION TO KINECT SENSOR

### *A. Kalman filtering model*

Kalman filter is an effective tool to produce optimal estimates using a series of noisy measurements [10]. In the case of Kinect sensor, disturbance comes from the unsteady capture of depth data. The filter has the advantage of working in real-time (at the same time of the capture) and uses only a small knowledge about the last state. We have already proven the adaptability of Kinect's data to this model (Kalman) in the previous section.

The usefulness of such adaptation is to improve the accuracy of moving robots tracking and obstacle avoidance

applications based upon consumer RGBD cameras. The work is motivated by the properties we already discovered about Kinect's depth data. The working principle of Kalman filter is based on a recursive **prediction** of the next state and its **correction**.
Linear Kalman filter equations in prediction-correction form are given below:

*Prediction:*

$$\overline{x_k} = Ax_{k-1} + Bu_k + w_k \quad (9)$$
$$\overline{P_k} = AP_{k-1}A^T + Q_k \quad (10)$$

*Correction:*

$$K_k = \overline{P_k}H^T(H\overline{P_k}H^T + R_k)^{-1} \quad (11)$$
$$x_k = \overline{x_k} + K_k(z_k - H\overline{x_k}) \quad (12)$$
$$P_k = (1 - K_kH)\overline{P_k} \quad (13)$$

Where: for each discrete time-step $\boldsymbol{k}$, $\overline{\boldsymbol{x_k}}$: state estimate; $\boldsymbol{z_k}$: the raw measured output; $\overline{\boldsymbol{P_k}}$: priori estimate error covariance; $\boldsymbol{K_k}$: Kalman gain; $\boldsymbol{A}$: the state-transition model; $\boldsymbol{B}$: the control-input model; $\boldsymbol{H}$: the observation model; $\boldsymbol{w_k}$: process noise, $\boldsymbol{v_k}$: measurement noise, $\boldsymbol{Q_k}$: the covariance of process noise; $\boldsymbol{R_k}$: the covariance of measurement noise and $\boldsymbol{u_k}$ : control signal. Process and observation noises should be independent, white and with normal distribution; $\boldsymbol{w_k \sim N(0, Q_k)}$ and $\boldsymbol{v_k \sim N(0, R_k)}$.

### B. Kalman filter on Kinect sensor

For a given pixel in a frame at time-step $\boldsymbol{k}$; $\overline{\boldsymbol{Z}}_{\boldsymbol{k}}$: is the estimate depth; $\widetilde{\boldsymbol{Z_k}}$ : measured depth; $\overline{\boldsymbol{P}}_{\boldsymbol{k}}$ : priori estimate error covariance and $\boldsymbol{K_k}$: kalman gain. We have a one to one correspondence between state/measurements, $\boldsymbol{H = I_3}$; $\boldsymbol{A}$: the state-transition model; $\boldsymbol{B = 0}$ : for fixed sensor or the control-input model of the moving robot; $\boldsymbol{R_k = \sigma_k^2}$: the covariance of the observation noise (related to the standard deviation, and differs from one Z-Level to another proportionally to the distance from the sensor). Thus, for sensor/scene configuration (Z normally evolves according to the dynamics of the scene defined by the state transition matrix $\boldsymbol{A}$. and the motion of the mobile robot if $\boldsymbol{B \neq 0}$), Kalman equations (9 to 13) become:
*Prediction:*

$$\overline{Z}_k = AZ_{k-1} + Bu_k \quad (14)$$
$$\overline{P}_k = AP_{k-1}A^T \quad (15)$$

*Correction:*

$$K_k = \overline{P}_k(\overline{P}_k + R_k)^{-1} \quad (16)$$
$$Z_k = \overline{Z}_k + K_k(\widetilde{Z_k} - \overline{Z}_k) \quad (17)$$
$$P_k = (1 - K_k)\overline{P}_k \quad (18)$$

As we can see from Figure 10, Kalman filter clearly attributes off-Z-Levels optimal coordinates to our discrete clusters of 3D points, and approaches them the best to their real world positions.

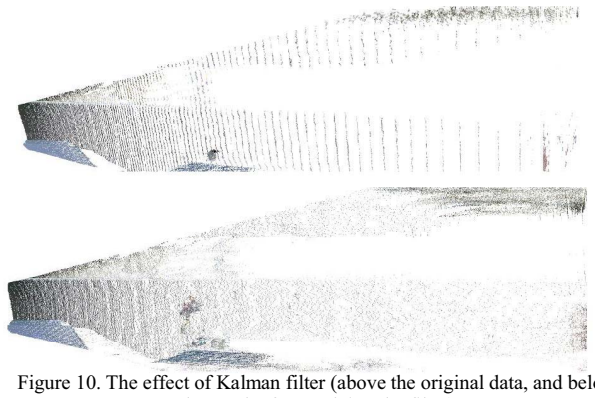

Figure 10. The effect of Kalman filter (above the original data, and below the result after applying the filter)

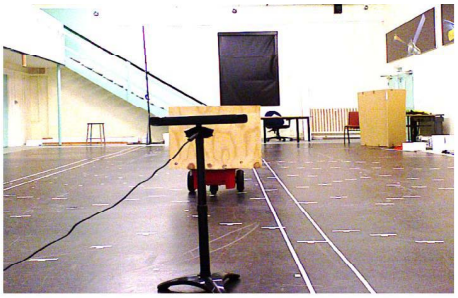

Figure 11. Sensor/Tracked panel setup

## V. EXPERIMENTS

### A. Experimental setup overview

To validate our findings, we conducted some experiments by tracking a moving flat object (the wooden panel carried by the robot in front of the camera Figure 11) in a typical indoor environment. We move the object around different known landmarks. At every point, we position the panel at a parallel direction in front of the sensor, and we grab the depth map of the scene. Afterwards, we extract the 3D coordinates of the object and we compare them to what we have already got from the ground truth data[3]. The most important measure is the range (distance between the sensor and the object). Once we have accurately determined this entity ($\boldsymbol{z_i}$), we can compute the two others ($\boldsymbol{x_i, y_i}$) using the 2D screen coordinates ($\mathbf{u_i}$ , $\mathbf{v_i}$) and the intrinsic parameters of the camera (6). This approach allows us to decide whether the theoretical concepts we developed earlier are experimentally valid. Figures (12), (13), (14) illustrate what we are expecting from the filter.

- Ground-truth data in black, with the sensor pointed toward the scene Figure 12.
- Raw tracking output of the sensor in red Figure 13.
- Corrected tracking in green Figure 14.

[3] BOSCH DLE 40 Professional Laser Rangefinder

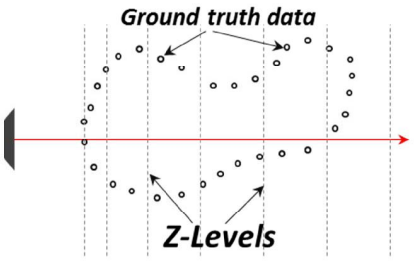


Figure 12. Experimental setup

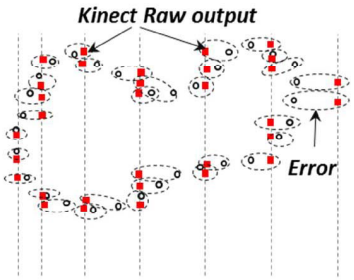


Figure 13. Kinect raw measurements

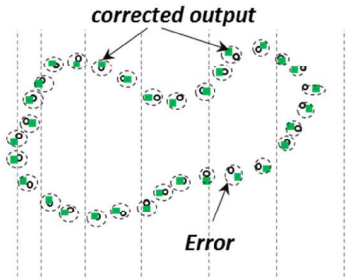


Figure 14. Corrected measurements

When the sensor captures a point cloud, the resulting 3D data are automatically distributed on the discrete Z-Levels. However, the original points' data come from the continuous real world. Their corresponding images in Kinect's space lie in the parallel planes.

The error in measurement is proportional to the gap between Z-Level Figure 13. Kalman filter takes these noisy raw data as input, optimises them, and consequently approaches the best their real world counterparts Figure 14.

We run our experiments with a fixed viewpoint and a moving target Figures 11. This setup is classified among tracking applications. Nevertheless, the concept remains applicable for a moving viewpoint (the camera is carried by a mobile robot).

The range of the tracked object evolves linearly over time according to our state-transition matrix $\boldsymbol{A}$. The system is typically Newtonian, but we are only interested in optimising the positions based on the depth output of Kinect sensor.

*B. Results*

We compared the corrected data generated by the filter against Kinect's raw output and another command data (these data are the elementary motion vectors between successive positions). Figure 15 shows the error graphs for all three approaches over time. Kalman filter clearly has the lowest error (0.016 m, in green). Whereas, Kinect marked an error of (0.072 m, in blue) and command data have the worst accuracy (0.1 m. in red). We tested the filter with a standard deviation $\boldsymbol{\sigma_k}$ where $\boldsymbol{i = 1, 2, 3, 4, 5, 6;}$ the best result was acquired with $\boldsymbol{i = 3}$. In other words, the sensor we used is more likely to have a Gaussian noise with a standard deviation of:

$$\sigma_k = ( Z_{k+3} - Z_{k-3}) / 2, \forall k \in I_k \quad (19)$$

Figure 16 illustrates the behaviour of estimation error covariance. From the graph, we notice that it is decreasing over time. This means that, the estimation is getting more and more accurate and the filter is working properly.

Figures (17), (18) show the results. Kinect accuracy is acceptable in a range below 3.5m from the sensor (Maximum error ≤ 5cm). However, when we exceed this barrier, the error increases dramatically up to 20cm at 8m. The actual operating range of the sensor is (0.6m to 3.5m) out of which Kinect is not meant to work correctly [2].

We remain the reader that one of the aims of this study is to extend the native operational range for Kinect sensor, using an extra filtering module which deals with the decreasing accuracy over growing range.

The results we got show that we could afford an extra 3.5m with no hardware improvement. The space to cover becomes broader, with a better accuracy (it is actually disproportional to the range, but with a small slope, compared to the raw measurements). More importantly, the filter works at the same time of the capture, and it does not affect the real-time asset of the sensor.

When we exceed the range of (7m), Kinect's output becomes very noisy and the filter could not handle these low quality measurements. This is a hardware limit which is out of the scope of this paper.

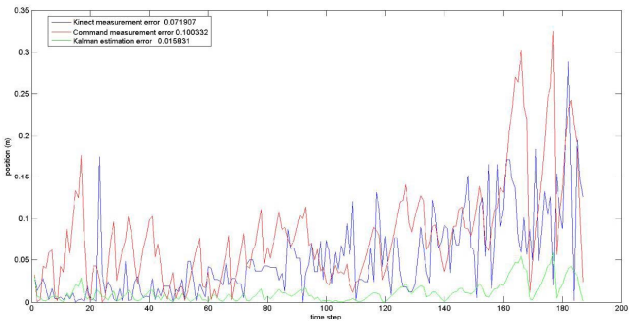


Figure 15. Kinect, Command and Kalman error graphs

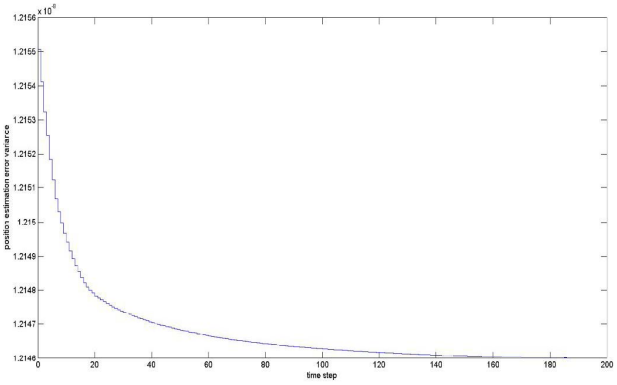


Figure 16. Error variance behaviour over time

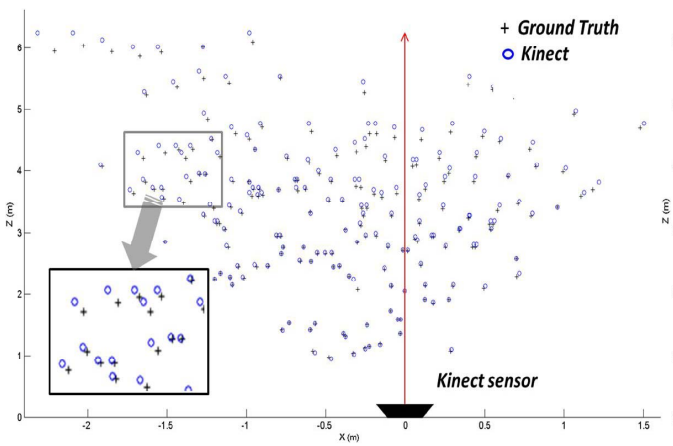


Figure 17. Raw Kinect experimental measurements

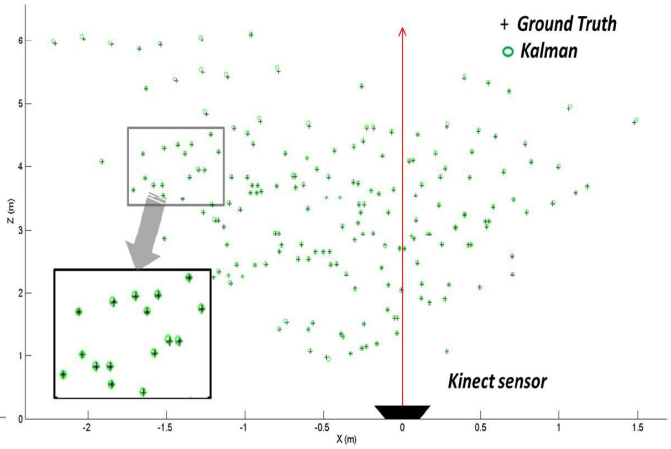


Figure 18. Kalman experimental correction

## VI. CONCLUSION AND FUTURE WORK

We presented a novel adaptation of Kalman filter to optimise the output of commodity RGBD camera (Kinect). This filtering stage works in real-time and does not slow down the frame rate of the sensor. We developed a thorough proof of our adaptation to Kinect sensor, with the corresponding analysis of the different properties characterising the camera's depth output. A detailed interpretation of the behaviour related to the depth map was presented along with the extraction of the parameters needed for filter ( $\sigma_k$).

We conducted a fixed viewpoint tracking scenario, with one Kinect in front of which evolves an object. The results prove the effectiveness of the filter with the appropriate parameters we found. Consequently, we extended the range where the camera could be accurate (from 3.5m for the raw output alone to 7m with filtered output). The RMSE errors issued from the experiments were 0.072m for Kinect sensor and 0.016m for the filtered data, with a maximum depth of 7m.

This adaptation is not only useful for tracking algorithms using Kinect as a rangefinder, but also for every solution based upon the sensor's depth output. The model remains the same, we just need to tune $\sigma_k$ according to the target application.

As future work, we aim to develop a co-operative tracking system that integrates more than one camera for a better covering of the environment and a more accurate output. The combination of more than one sensor's data has the benefit of taking the best from every element, and consequently improving the accuracy of the overall system.

## REFERENCES

[1] Schulz, D.; Burgard, W.; Fox, D.; Cremers, A.B., "Tracking multiple moving objects with a mobile robot," Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on , vol.1, no., pp.I-371,I-377 vol.1, 2001.

[2] Jing Tong; Jin Zhou; Ligang Liu; Zhigeng Pan; Hao Yan, "Scanning 3D Full Human Bodies Using Kinects," Visualization and Computer Graphics, IEEE Transactions on , vol.18, no.4, pp.643,650, April 2012.

[3] Colombo, C.; Allotta, B., "Image-based robot task planning and control using a compact visual representation," Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on , vol.29, no.1, pp.92,100, Jan 1999.

[4] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unknown indoor environments," International Journal of Micro Air Vehicles, 2009.

[5] Endres, F.; Hess, J.; Engelhard, N.; Sturm, J.; Cremers, D.; Burgard, W., "An evaluation of the RGB-D SLAM system," Robotics and Automation (ICRA), 2012 IEEE International Conference on , vol., no., pp.1691,1696, 14-18 May 2012.

[6] Ruhnke, M.; Kummerle, R.; Grisetti, G.; Burgard, W., "Highly accurate 3D surface models by sparse surface adjustment," Robotics and Automation (ICRA), 2012 IEEE International Conference on , vol., no., pp.751,757, 14-18 May 2012.

[7] Valenti, Bethke, et al, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery", 2006.

[8] Y. Yoshihata, K. Watanabe, Y. Iwatani and K. Hashimoto,Multi-Camera Visual Servoing of a Micro Helicopter Under Occlusions", book edited by Rong-Fong Fung, ISBN 978-953-307-095-7, Published: April 1, 2010.

[9] K. Khoshelham, "Accuracy analysis for Kinect depth data", ITC Faculty of Geo-information Science and Earth Observation, University of Twente, 2011.

[10] R.E. Kalman, "A new approach to linear filtering and prediction problems". Journal of Basic Engineering 82 (1), pp. 35-45. 1960.

[11] Liebowitz, D.; Zisserman, A., "Metric rectification for perspective images of planes," Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on , vol., no., pp.482,488, 23-25 Jun 1998.