Stable six-DoF head-pose tracking in assistive technology application

# Stable Six-DoF Head-Pose Tracking in Assistive Technology Application

Edwin Walsh*‡, Walter Daems*, Jan Steckel*‡
*FTI - Cosys Lab, University of Antwerp, Belgium
*Centre for Care Technology, University of Antwerp, Belgium
‡Flanders Make Strategic Research Centre, Belgium
edwin.walsh@UAntwerpen.be

***Abstract*—This paper describes an alternative approach to giving people with limited hand and arm movement the ability to select objects on a computing device using head movement (head mouse). To allow for better filtering of unintentional head movement, and to allow for a faster update rate of the head mouse, the full six-DoF pose of the head is estimated using a low-cost camera and IR markers, a 3-axis accelerometer, and a 3-axis magnetometer. The pose estimation problem was cast in a probabilistic fashion in which information from the different sensors is fused into a single a-posteriori distribution for the sensor pose. Simulations were run to analyze the influence of the proposed sensor fusion algorithm on the stability and accuracy of the pose estimation, and thus on the ability to point a mouse cursor to a specified location on a screen. Experiments were then performed validating the results from the simulations on real sensors. The proposed algorithm was shown to give more accurate and stable results than using only a camera to estimate the six-DoF pose.**

## I. Introduction

Life with a disability is often challenging, but with help and support, a large proportion of disabled people can learn to cope with their disability and face the challenge head on. An important factor in dealing with a disability is the continuing struggle for independence [1]. By reducing the effort needed to control certain basic interactions, the self-reliance of people with disabilities can be greatly increased [2]. These basic interactions include tasks like operating light switches, electrical doors and windows, television, etc., but also operating personal computing devices (PC, tablet, smartphone).

For most of these interactions a solution already exists, but all of them have their own issues (e.g. too slow, inaccurate, requiring clutching, etc.), none of them giving a solution that encompasses the entire problem. The focus of our research is targeted at people with limited arm and finger mobility and control who retain sufficient head mobility. Typical examples are people affected by amyotrophic lateral sclerosis (ALS) [3] or other quadriplegia.

While it is our goal to provide a solution for all interactions with home automation systems and computing devices, this paper will focus on the ability to select objects on a computing device, or simply put, the ability to control a mouse cursor through head movement, which we will further refer to as a head mouse.

### *A. State of the art*

Although the target group is relatively small, a large number of head mice have already been provided. These head mice can be classified based on the order of controls [4], i.e. first-order control (velocity-control (1)) and zero-order control (position-control). Position-control can be subdivided into relative position control (2) (where displacement is measured) and absolute position control (3). Examples from each of these classifications are (1) joysticks, (2) touch pads, and (3) touch screens. A more detailed comparison of these systems is made by the authors in [5]. What can be concluded from the previously mentioned paper is that using absolute position control has the advantage of being fast, and not requiring clutching. Clutching is a break in control-to-display mapping when the input device can be moved independently of the cursor (e.g. lifting the finger to reposition it on a trackpad) [6].

The solution provided in [5] makes use of a low-cost camera attached near the head (e.g. on a pair of glasses), and IR-LED markers placed near the screen at known positions. Using a homography matrix on the detected led pixel coordinates, the intersection of the principal axis of the camera and the computer screen can be calculated, and the mouse cursor is placed on this location. This initial prototype however still has a few issues. The refresh rate of the mouse cursor coordinate is limited by the refresh rate of the camera, which for low-cost camera's typically is 30 Hz. This is too slow to allow for a smooth control of the mouse cursor [7]. Another issue is that due to the precision of the system, minimal head tremors will cause the mouse cursor to jitter. A non-linear tremor filter (Gaussian attractor) was provided in [5], but it was not sufficiently effective to allow the selection of very small objects on the computer screen.

To overcome these issues, we propose a solution where information from multiple sensors is fused to allow for a faster refresh rate. To allow for easy sensor fusion, the full 6-DoF pose of the camera is calculated. Although the main focus of this paper is based on the ability to control a mouse cursor, knowing the full 6-DoF pose allows for applications like posture detection, gaming interfaces, etc. It should also be noted that, although the goal of the proposed solution is to be able to filter user tremors, the actual tremor filtering will not be part of this paper.

## II. Proposed System

We start off from a low-cost short-wave infrared camera, with four IR LEDs placed around the screen [5]. In addition a 3-axis accelerometer and 3-axis magnetometer are added. Since pose estimation using only a camera is very sensitive to measurement errors due to the ill-posed nature of the inverse problem, we propose using a probabilistic approach for estimating the pose of the system. This approach has the advantage of being more robust against outliers (among the measurements) when compared to a deterministic method, and is also better at handling missing or incomplete measurements, due to allowing for the use of recursive Bayesian filtering techniques such as particle filters and extended Kalman filters. We therefore cast the pose estimation problem in a probabilistic fashion in which information from different sensors is fused into a single a-posteriori distribution for the sensor pose. Each sensor can present its sensor data as it becomes available and that data will be used to refine the estimated pose. We define the sensor's pose $\vec{P}$ as:

$$\vec{P} = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix}^T \tag{1}$$

where the rotational component $(\alpha, \beta, \gamma)$ stands for the Euler angles $ZY'X''$ with $\gamma$ the rotation around the Z-axis, $\beta$ the rotation around the new Y-axis, and $\alpha$ the rotation around the new X-axis.

### *A. Camera*

We define the likelihood of a camera measurement $\mathbf{M}_c$ given a certain pose $\vec{P}$ as

$$\mathcal{L}_c(\mathbf{M}_c|\vec{P}) = exp(-\frac{1}{2} \cdot (\vec{M}_c - \vec{C}_c)^T \cdot \mathbf{\Sigma}_c^{-1} \cdot (\vec{M}_c - \vec{C}_c)) \tag{2}$$

where $\vec{M}_c$ is the vectorized version of the matrix $\mathbf{M}_c$ which defines the sorted coordinates of the blobs created by the $n$ markers on the camera image:

$$\mathbf{M}_c = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y1 & y_2 & \dots & y_n \end{bmatrix}^T \tag{3}$$

and $\vec{C}_c$ is the vectorized version of the calculation matrix $\mathbf{C}_c$ which represents the calculated blob coordinates given a certain pose

$$\mathbf{C}_c = \begin{bmatrix} x_{c1} & x_{c2} & \dots & x_{cn} \\ y_{c1} & y_{c2} & \dots & y_{cn} \end{bmatrix}^T \tag{4}$$

The covariance matrix $\mathbf{\Sigma}$ is set to be a scaled identity matrix, since we assume normally-distributed independent measurements:

$$\mathbf{\Sigma}_c = \sigma_c \cdot \mathbf{I}_{2n \times 2n} \tag{5}$$

with $\sigma_c$ experimentally set to 10 pixels. For realistic results, each measured blob center must be compared to the correct corresponding projected marker. This was done by putting the markers in a predefined pattern from which the individual markers can be deducted. Another possible way of identifying each marker would be by modulating each marker in a unique manner, e.g. with simple On-off keying modulation.

To calculate $\mathbf{C}_c$ we need a model of the camera to transform the marker positions in world coordinates into projected points onto the camera sensor. This can be achieved using the following equation defined in [8]

$$\mathbf{C}_c^T = \mathbf{K} \cdot \mathbf{R} \cdot [\mathbf{I}_{3\times3}| - \tilde{C}] \cdot \mathbf{X}_w \tag{6}$$

with $\mathbf{X}_w$ a $4 \times n$ matrix containing the marker positions in homogeneous coordinates, $\tilde{C}$ the coordinates of the camera center in the world coordinate frame, and $R$ a $3 \times 3$ rotation matrix representing the orientation of the camera coordinate frame. The vertical line $|$ denotes matrix concatenation. The matrix $\mathbf{K}$ is called the camera calibration matrix and is defined as

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{7}$$

where $\alpha_x$ and $\alpha_x$ define the focal distance in pixel coordinates, $x_0$ and $y_0$ define the principal point, and $s$ is the skew factor. To make the modeled data more realistic, non-linear radial and tangential lens distortion is then applied to the calculated camera coordinates.

The camera intrinsics and the radial and tangential lens distortion parameters can be estimated using the Matlab Image Calibration Toolbox [9]. However, due to the IR bandpass filter placed in front of the camera used herein, which blocks the view of a checkerboard pattern required by the calibration toolbox, a ledboard was created with 30 IR LEDs equally spaced in a 6 by 5 grid which can be seen in Fig. 1. The detected blob centers could then be used as input for the camera-calibration functions.

### *B. Accelerometer and magnetometer*

We define the likelihood of the accelerometer and magnetometer measurements in a similar fashion as the camera measurements

$$\mathcal{L}_a(\vec{M}_a|\vec{P}) = exp(-\frac{1}{2} \cdot (\vec{M}_a - \vec{C}_a)^T \cdot \mathbf{\Sigma}_a^{-1} \cdot (\vec{M}_a - \vec{C}_a)) \tag{8}$$

$$\mathcal{L}_m(\vec{M}_m|\vec{P}) = exp(-\frac{1}{2} \cdot (\vec{M}_m - \vec{C}_m)^T \cdot \mathbf{\Sigma}_m^{-1} \cdot (\vec{M}_m - \vec{C}_m)) \tag{9}$$

with $\vec{M}_a = \begin{bmatrix} x_a & y_a & z_a \end{bmatrix}^T$ and $\vec{M}_m = \begin{bmatrix} x_m & y_m & z_m \end{bmatrix}^T$ the measurements from the respective three-axis accelerometer and magnetometer, and $\vec{C}_a = \begin{bmatrix} x_{ca} & y_{ca} & z_{ca} \end{bmatrix}^T$ and $\vec{C}_m = \begin{bmatrix} x_{cm} & y_{cm} & z_{cm} \end{bmatrix}^T$ the calculated values from the respective accelerometer and magnetometer models for a certain pose $\vec{P}$. The covariance matrix $\mathbf{\Sigma}$ is again set to be a scaled identity matrix:

$$\mathbf{\Sigma}_a = \sigma_a \cdot I \tag{10}$$

$$\mathbf{\Sigma}_m = \sigma_m \cdot I \tag{11}$$

with $\sigma_a$ experimentally set to $6\,\mathrm{mg}$ and $\sigma_m$ experimentally set to $10\,\mathrm{mT}$.

The accelerometer and magnetometer models are much simpler, and simply transpose the corresponding gravity and magnetic field vector according to the rotational component $(\alpha, \beta, \gamma)$ from the pose $\vec{P}$.

$$C_a = \mathbf{R} \cdot \vec{G} \tag{12}$$

$$C_m = \mathbf{R} \cdot \vec{B} \tag{13}$$

where $\vec{G} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, $\mathbf{R}$ is the rotation matrix to the corresponding Euler angles $(\alpha, \beta, \gamma)$, and $\vec{B}$ is defined by the local magnetic field. The magnetic field can be calculated for the assumed position using a model of the Earth magnetic field (e.g. using the NCEI Geomagnetic Calculators [10]). The time-varying accelerometer biases and magnetometer biases where neglected in this model. It is the authors' view that these will only have a minor effect on the variance of the estimated pose. This however requires further investigation.

### C. Sensor fusion

The likelihood from the combined measurements $\mathbf{M}$ can then be defined as

$$\mathcal{L}(\mathbf{M}|\vec{P}) = \mathcal{L}_c \cdot \mathcal{L}_a \cdot \mathcal{L}_m \tag{14}$$

We can derive a posterior probability function for the pose $\vec{P}$ given measurement $M$ using Bayes rule:

$$P(\vec{P}|M) = \frac{\mathcal{L}(M|\vec{P}) \cdot P(\vec{P})}{P(M)} \tag{15}$$

with $P(\vec{P})$ the prior distribution for the pose $\vec{P}$ and $P(M)$ the marginal distribution of the product in the numerator. The prior $P(\vec{P})$ allows constraining the solution space to sensible poses (e.g. the sensors will never be behind the IR LED markers). We will minimize the negative logarithm of the posterior distribution to arrive at a pose estimate for the sensors:

$$\vec{P}_{est} = argmin(-log(P(\vec{P}|M)) \tag{16}$$

The posterior is minimized using an unconstrained non-linear minimization method (Nelder-Mead).

## III. Experimental results

To analyze the proposed algorithm we performed extensive simulations. First we looked at the influence of noise on the effectiveness of the pose estimation, then we analyzed the influence of the sensor locations. Both experiments were performed with and without sensor fusion. To analyze the effectiveness of the algorithm on controlling a mouse cursor, we simulated pointing the camera at the screen for 441 coordinates evenly spread out between the markers, from the same locations as the previous simulation. We then proceeded with some real-life experiments where the sensors were mounted on a robot arm, to validate the effectiveness of the algorithm is a real-life application.

The coordinate system used throughout this paper is based on a world coordinate system with the origin at the center of the base of the robot arm as depicted in Fig. 1.

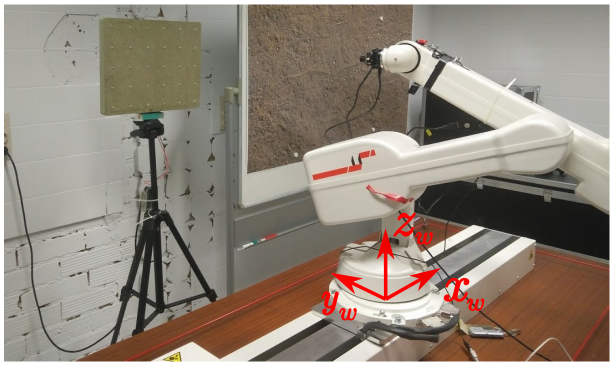

Fig. 1. Example of the calibration setup where the camera was placed on a ST Robotics R17 robot arm.

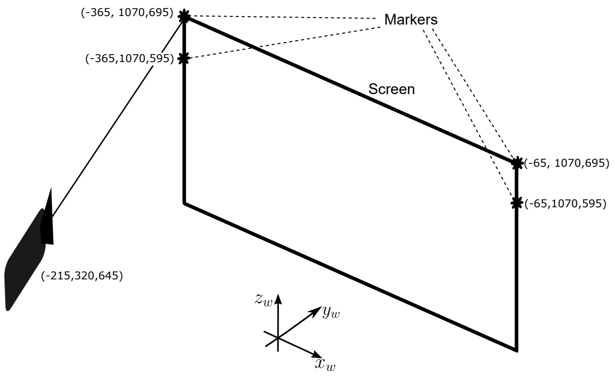

Fig. 2. Camera and marker positions in world coordinate system as defined in Fig. 1

### A. Simulations

To simulate sensor measurements, we built a model for each of the sensors to which we can add zero-mean Gaussian noise. To define a base noise level for the sensors, 10000 measurements were taken with each of the sensors from which a standard deviation of the Gaussian noise was deducted ($\sigma_c = 0.1\,\text{px}, \sigma_a = 0.78\,\text{mg}, \sigma_c = 2.2\,\text{mT}$). These measurements were also used to define the magnetic field vector ($\vec{B} = \begin{bmatrix} -0.0755 & -0.1617 & -0.4615 \end{bmatrix}$) which was validated to have a similar inclination as calculated by the NCEI Geomagnetic Calculators [10]. To analyze the effect of noise on the results, we simulated 100 measurements for each of 30 logarithmically spaced noise levels. An arbitrarily defined pose and location of the markers where chosen as depicted in Fig. 2.

Fig. 3 shows the simulation results for varying values of camera noise by either using only the camera, or using sensor fusion, for estimating the pose $\vec{P}$. The experimentally defined base noise levels were added to the other sensors. One of the major issues when using a single camera to calculate the pose of the camera is the ill-posedness of the inverse problem. This is due to ambiguities resulting from different camera poses generating a very similar projected image. This effect can be noticed in Fig. 3.c and Fig. 3.d where the standard deviation of the estimated pose is plotted. It can also be seen

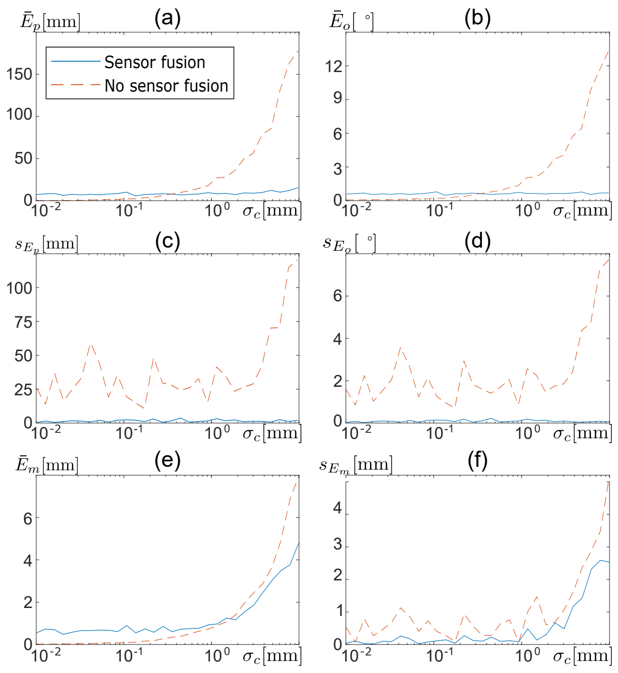

Fig. 3. Simulation results for varying values of camera noise by using either only the camera, or sensor fusion, for estimating the pose $\vec{P}$. 100 measurements were taken for each varying value of camera noise. The true pose of the camera is defined in Fig. 2. (a) Mean value of the position error. (b) Mean value of the orientation error, calculated as the angular value of the axis-angle transformation between the true rotation and the estimated rotation. (c) Standard deviation of the position error. (d) Standard deviation of the orientation error. (e) Mean value of the target-coordinate estimation error, calculated as the intersection between the principal axis of the camera and the plane created by the marker coordinates. (f) Standard deviation of the target-coordinate estimation error.

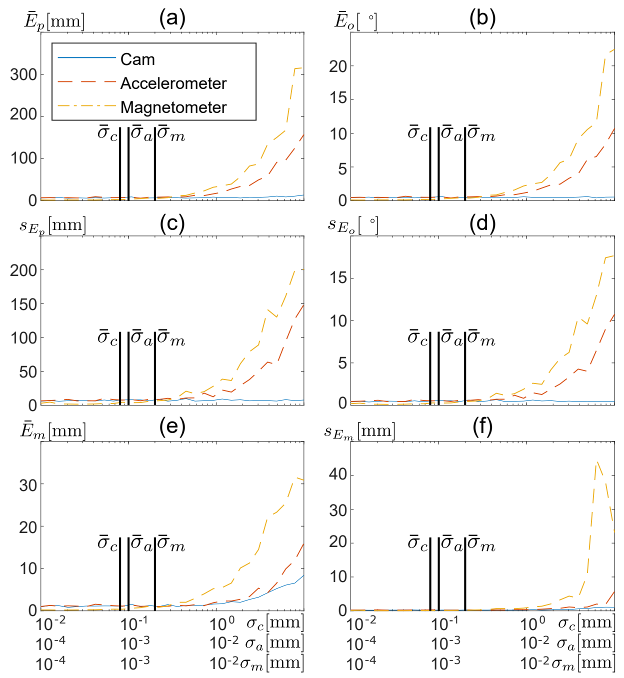

Fig. 4. Simulation results for varying values of either camera noise, accelerometer noise, or magnetometer noise, all using sensor fusion for estimating the pose $\vec{P}$. 100 measurements were taken for each varying value of sensor noise. The true pose of the camera is defined in Fig. 2. $\bar{\sigma}_c$, $\bar{\sigma}_a$, and $\bar{\sigma}_m$ signify the mean measured values of the respective sensors. (a) Mean value of the position estimation error. (b) Mean value of the rotation estimation error, calculated as the angular value of the axis-angle transformation between the true rotation and the estimated rotation. (c) Standard deviation of the position estimation error. (d) Standard deviation of the rotation estimation error. (e) Mean value of the target-coordinate estimation error, calculated as the intersection between the principal axis of the camera and the plane created by the marker coordinates. (f) Standard deviation of the target-coordinate estimation error.

that the sensor fusion algorithm greatly reduces the number of ambiguities leading to a much smaller standard deviation of estimated poses. However, due to accelerometer and magnetometer noise, the mean estimation error is somewhat larger for very small noise values on the camera measurements.

Since the goal of this paper is to analyze the effectiveness of the algorithm to calculate a mouse coordinate, Fig. 3.e and Fig. 3.f depict the mean and standard deviation of the difference between the true target coordinate, and the intersection between the camera's principal axis and the plane formed by the marker coordinates. Both the camera and sensor fusion systems perform reasonably well, even for high values of the camera noise. It is however clear that there is still room for improvement, which could be achieved by a recursive Bayesian filter which includes a motion model and sensor data from a gyroscopic sensor.

Fig. 4 shows the simulation results for varying values of camera noise, accelerometer noise, and magnetometer noise, all using sensor fusion for estimating the pose $\vec{P}$. It is clear from Fig. 4.a-d that high magnetometer noise has the biggest impact on the performance of the sensor fusion algorithm. However, using noise values near the measured noise values of the prototype, gives very satisfactory results for the pose estimation. The same can be concluded for the estimated

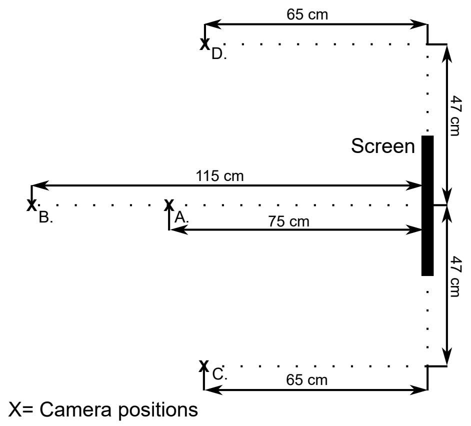

Fig. 5. Camera positions for the simulated and real-life experiments.

intersection point.

To analyze the influence of the camera position on the algorithm, we simulated camera positions in analogy to [5], as depicted in Fig. 5. For each of these positions 200 simulations were performed. First we used only the camera to estimate the pose, the results of which can be found in Fig. 6.a-b. The same simulations were then run using sensor fusion, the results

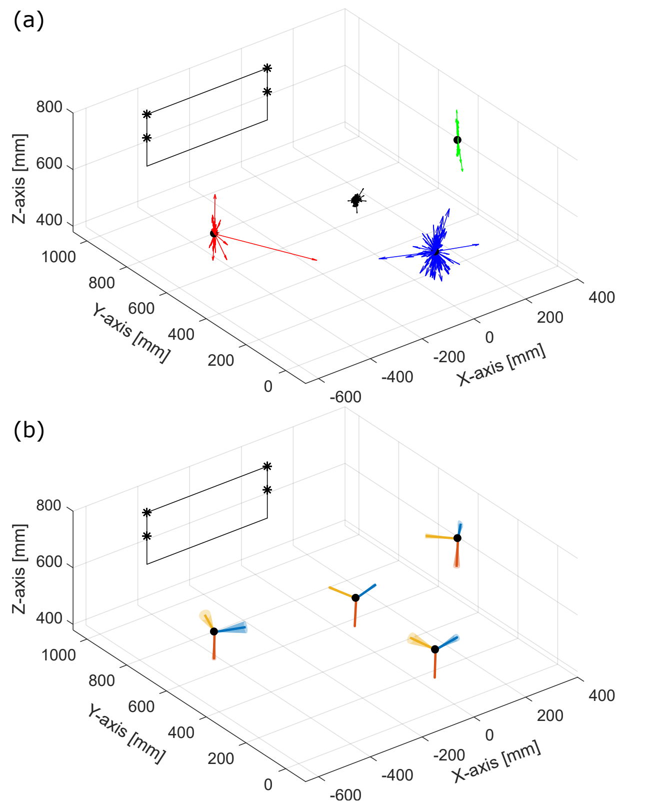

Fig. 6. The pose estimation error using only the camera measurements for the four locations defined in Fig. 5. The camera was aimed at the top left marker in all cases. (a) Quiver plot of the estimated position errors. The arrows are of true size with respect to the axis scale. (b) Standard deviation of the rotation estimation error, calculated as the angular value of the axis-angle transformation between the true rotation and the estimated rotation.

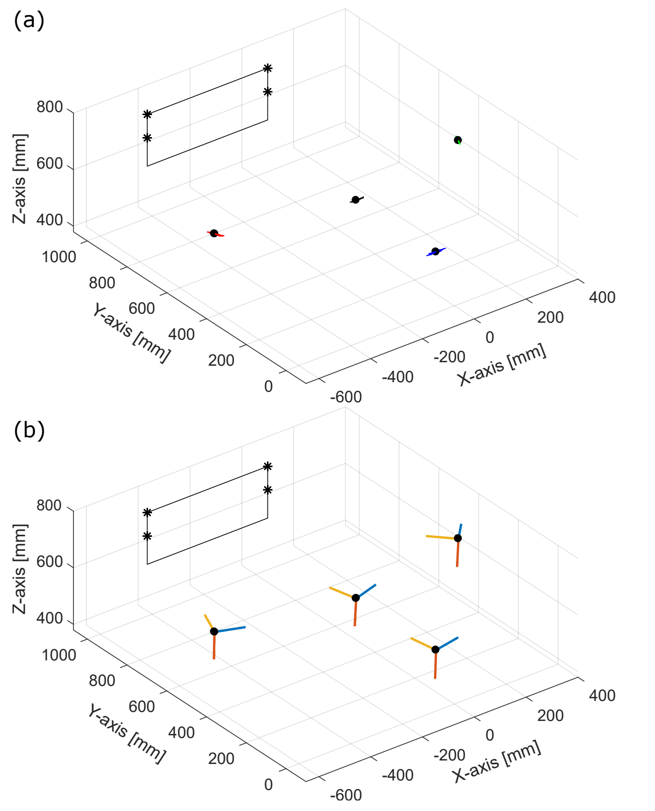

Fig. 7. The pose estimation error using sensor fusion for the four locations defined in Fig. 5. The camera was aimed at the top left marker in all cases. (a) Quiver plot of the estimated position errors. The arrows are of true size with respect to the axis scale. (b) Standard deviation of the rotation estimation error, calculated as the angular value of the axis-angle transformation between the true rotation and the estimated rotation.

of which can be found in Fig. 7.a-b. These figures show the pose estimation error for the four locations defined in Fig. 5. The camera was aimed at the top left marker in all cases. Comparing Fig. 6 with Fig. 7 clearly shows the impact of the sensor fusion algorithm on the effectiveness of estimating the camera pose.

Since this sensor fusion algorithm will be used in an Assistive Technology application meant to control a mouse cursor, a simulation was run pointing the camera at 441 coordinates equally distributed over an area mimicking a screen. The intersection point between the camera's principal axis and the plane created by the markers was calculated, based on the proposed algorithm for 100 measurements per coordinate, the results of which can be found in Fig. 8. The top row of Fig. 8.a shows the mean error of the calculated intersection point for each target coordinate, the bottom row shows the standard deviation of the calculated intersection point for each target coordinate. While the mean value of the error is included for completeness, it is the authors' view that a relatively constant, smoothly changing bias error is of lower importance due to users automatically correcting for this bias due to seeing the screen cursor position [5]. A more important parameter is the standard deviation. The standard deviation is small for most points on the simulated screen, with a couple of outliers when the camera is positioned to the far left and right of the screen. These outliers in the standard deviation are caused by outliers in the calculated target coordinate. This is shown in Fig. 8.b, which shows a boxplot of 1000 new target coordinate calculations from the pose with the highest standard deviation from position D. It can be seen that for a small number of target coordinate calculations, the estimated target coordinate is very inaccurate. It is the authors' view that these outliers are caused due to numerical instability of the posterior minimization. This issue could be solved by introducing recursive Bayesian filter techniques which include a motion model and sensor data from a gyroscopic sensor.

## *B. Experiments*

To validate the proposed algorithm in a real-life test case we mounted the sensors on an ST Robotics R17 robot arm, allowing us to repeatably put the sensors in a specific pose. We performed the same accuracy test for the poses similar to the simulated test. The exact same poses were unreachable due to the limited range of the robot arm. The results of this experiment can be found in Fig. 9

While the range of the pose estimation error is slightly larger for the real measurements, compared to the simulated measure-

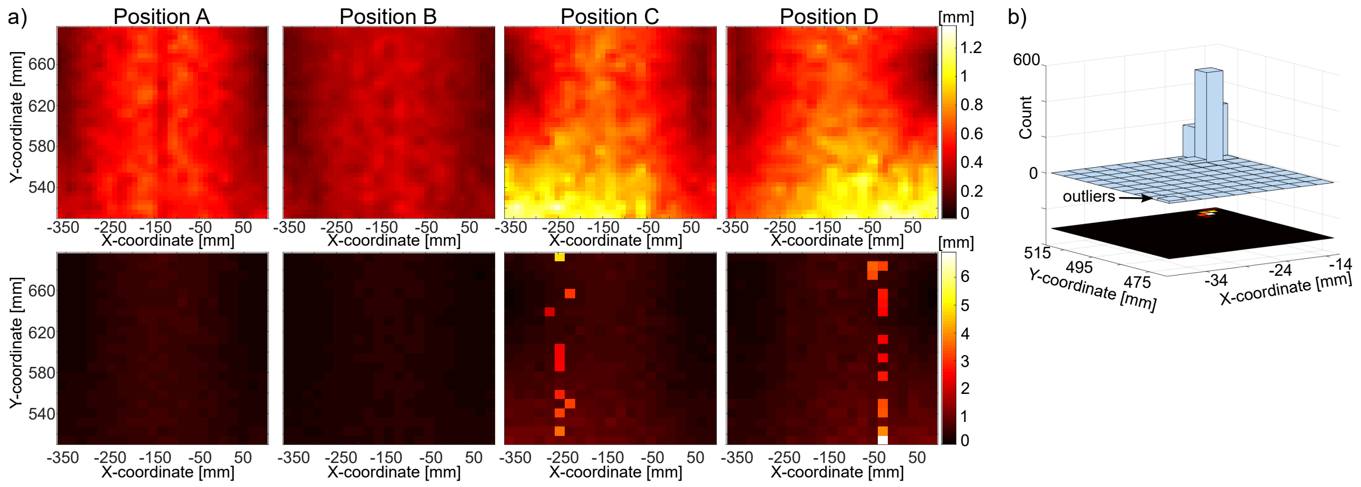

Fig. 8. (a) Result obtained by pointing the camera from four different locations (see Fig. 5.b) at 441 coordinates equally distributed over the area mimicking a 21inch 1920 × 1080 screen, and calculating the intersection point between the camera principal axis and the plane created by the markers, based on the proposed algorithm for 100 measurements per coordinate. The top row shows the mean error of the calculated intersection point for each target coordinate, the bottom row shows the standard deviation of the calculated intersection point for each target coordinate. b) Histogram of 1000 estimated cursor positions for the pose with the highest standard deviation from Fig. 8.a.

ments, it is still a very similar and satisfactory result. Mainly the positional error along the Z-axis, and the corresponding orientation error along the Y-axis are significantly larger than during simulation. This would indicate that the estimated noise levels for the accelerometer were estimated a little too low. This is confirmed when validating the standard deviation of the accelerometer during the real measurements, which came to a mean value of $2\,\mathrm{mg}$, compared to the value of $0.78\,\mathrm{mg}$ used in the simulations.

To gain a better understanding of the spread of the estimated pose values, a boxplot is shown in Fig. 10. The boxplots show that although the spread of the estimated pose values is relatively large, most of the estimated values are grouped really close to each other (the Q1, median, and Q3 lines are drawn on top of each other).

While accurate pose estimation is desirable for the filtering of the actual head pose movement, the main objective is still to be able to accurately point the mouse cursor on the screen. The accuracy of the calculation of the intersection point between the principal axis of the camera and the plane formed by the markers is visualized in Fig. 11. The calculated intersection points for the 100 measurements from each of the tested locations are plotted on top of a small icon of 16x16 pixels (=3.9667mm for a 21inch screen with resolution 1920x1080). It is clear that the calculated intersection point is accurate enough to allow for the selection of very small objects on a computer screen.

## IV. Conclusion and Future Work

In the introduction we explained the need for stable Six-DoF head-pose estimation for a specific Assistive Technology application. We provided a proposed Bayesian sensor-fusion algorithm for pose estimation based on the maximization of a posterior probability function. This algorithm was extensively tested using both simulated and real-life experiments. The

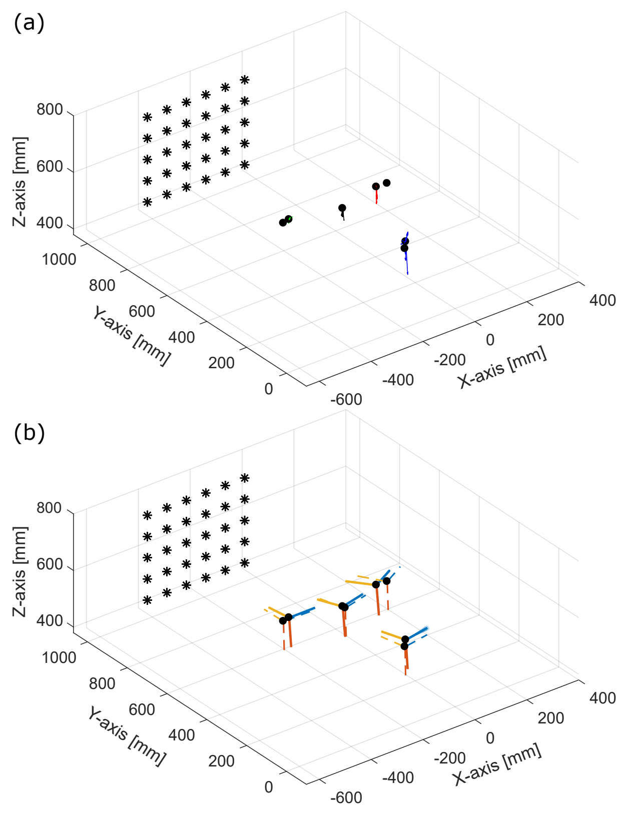

Fig. 9. Visualization of the pose estimation error for 100 measurements per location, using sensor fusion of real sensors. The camera was aimed at the top left marker in all locations. (a) Quiver plot of the estimated position errors. The arrows are of true size with respect to the axis scale. (b) Standard deviation of the rotation estimation error, calculated as the angular value of the axis-angle transformation between the true rotation and the estimated rotation.

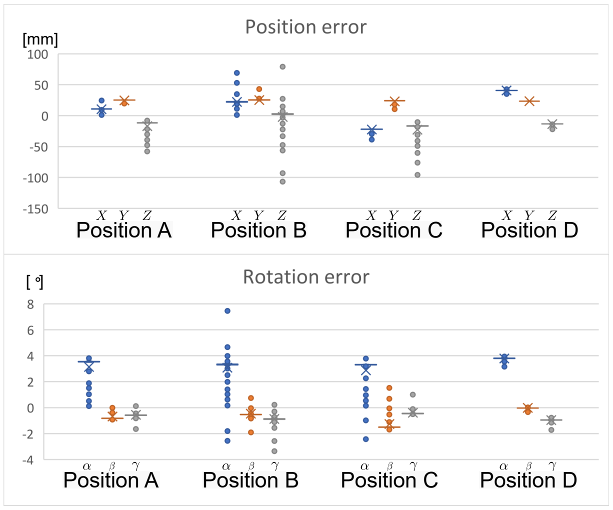

Fig. 10. Boxplot of the pose estimation error for 100 measurements per location, using sensor fusion of real sensors for the same locations as in Fig. 9. The boxplots are grouped per location and display the positional estimation error for the X, Y, and Z coordinate, and the orientation estimation error for the $\alpha$, $\beta$, and $\gamma$ angles.Note the Q1, median, and Q3 lines are drawn on top of each other

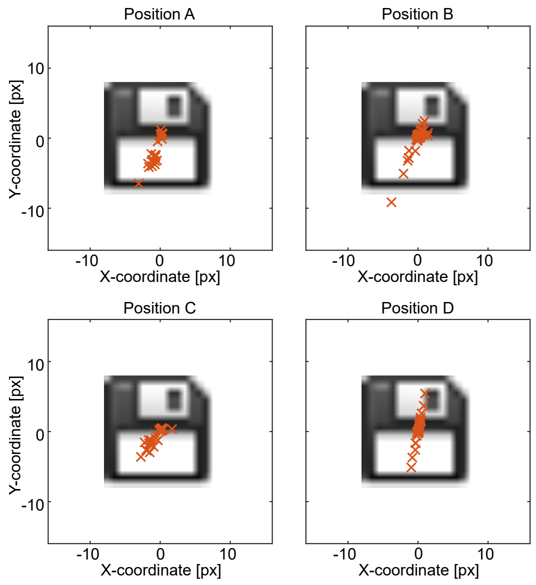

Fig. 11. Visualization of the calculated intersection points for 100 measurements from the locations defined in Fig. 5. The camera was aimed at the top left marker in all cases. An small icon of 16x16 pixels (=3.9667mm for a common 21inch screen with resolution 1920x1080 [11]) was added for clarity.

probabilistic approach to sensor fusion proves quite promising for the specific use-case of controlling a mouse cursor through head movement.

In future work the current posterior minimization using an unconstrained non-linear minimization method (Nelder-Mead), will be replaced by recursive Bayesian filtering techniques which include a motion model for the estimated head movement, and a better estimation of the prior distribution of the system states. Furthermore a gyroscopic sensor will be added for more stable measurements over short time frames to counteract the jitter caused by the accelerometer and magnetometer readings.

## ACKNOWLEDGMENT

This work is being supported by the Flemish Government (VAPH) as a part of the Flanders' care *Serenade* project.

## REFERENCES

[1] M. A. Nosek and M. J. Fuhrer, "Independence among people with disabilities: I. A heuristic model." *Rehabilitation Counseling Bulletin*, 1992.

[2] J. Soar and P. R. Croll, "Assistive technologies for the frail elderly, chronic illness sufferers and people with disabilities–a case study of the development of a Smart Home," 2007.

[3] L. P. Rowland and N. A. Shneider, "Amyotrophic lateral sclerosis," *New England Journal of Medicine*, vol. 344, no. 22, pp. 1688–1700, 2001.

[4] S. Zhai, "Human performance in six degree of freedom input control human performance in six degree of freedom input control," Ph.D. dissertation, University of Toronto, 1995.

[5] E. Walsh, W. Daems, and J. Steckel, "Assistive Pointing Device Based on a Head-Mounted Camera," *IEEE Transactions on Human-Machine Systems*, vol. PP, no. 99, pp. 1–8, 2017.

[6] M. Nancel, D. Vogel, and E. Lank, "Clutching is not (necessarily) the enemy," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 4199–4202.

[7] I. S. MacKenzie, "Input devices and interaction techniques for advanced computing," *Virtual environments and advanced interface design*, pp. 437–470, 1995.

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[9] *MATLAB version 9.2.0.538062 (R2017a)*, The Mathworks, Inc., Natick, Massachusetts, 2017.

[10] N. C. for Environmental Information, "Ncei geomagnetic calculators," 2017. [Online]. Available: https://www.ngdc.noaa.gov/geomag-web/

[11] W3Schools, "Browser display statistics," 2017. [Online]. Available: https://www.w3schools.com/browsers/browsers_display.asp