Correcting Drift, Head and Body Misalignments between Virtual and Real Humans

Vitor Reus, Márcio Mello, Luciana Nedel, Anderson Maciel
Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre, Brazil
{vureus,marcio.mello,nedel,amaciel}@inf.ufrgs.br

***Abstract*—Head-mounted displays (HMD) allow a personal and immersive viewing of virtual environments, and can be used with almost any desktop computer. Most HMDs have inertial sensors embedded for tracking the user head rotations. These low-cost sensors have high quality and availability. However, even if they are very sensitive and precise, inertial sensors work with incremental information, easily introducing errors in the system. The most relevant is that head tracking suffers from drifting. In this paper we present important limitations that still prevent the wide use of inertial sensors for tracking. For instance, to compensate for the drifting, users of HMD-based immersive VEs move away from their suitable pose. We also propose a software solution for two problems: prevent the occurrence of drifting in incremental sensors, and avoid the user from move its body in relation to another tracking system that uses absolute sensors (e.g. MS Kinect). We analyze and evaluate our solutions experimentally, including user tests. Results show that our comfortable pose function is effective on eliminating drifting, and that it can be inverted and applied also to prevent the user from moving their body away of the absolute sensor range. The efficiency and accuracy of this method makes it suitable for a number of applications in immersive VR.**

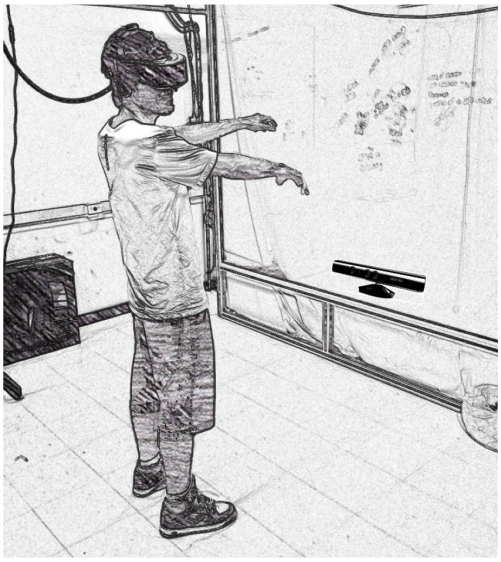
***Keywords*-virtual reality; tracking filters; inertial navigation; interactive computing**

I. Introduction

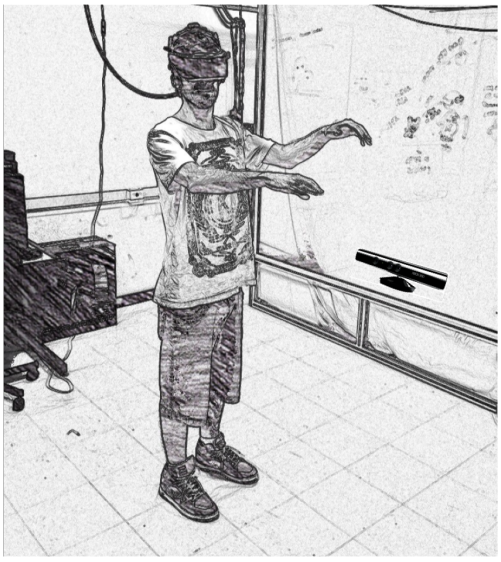
Virtual Reality (VR) systems often require a position tracking infrastructure to allow user navigation and interaction in a virtual space. Expensive and complex tracking systems are commercially available to track head position and orientation as well as limbs movements in real time. The information collected from such systems are then used to reproduce and apply user movements on their respective avatar for display or interaction.

In most cases, however, especially in research systems, such tracking is limited due to constraints as: funding (precise position trackers are expensive); availability of physical space; or the use of markers and complicated setups. Many immersive VR systems use a typical setup with a head mounted display (HMD) and some handheld interaction device, e.g. gamepads, magic wands, or specific VR controllers, which do not require complex tracking systems. Notice that the typical HMDs provide embedded inertial sensors for tracking of the head rotations. Handheld devices, in turn, may or may not require some tracking infrastructure. When they do, it is often simple as only one reference frame must be tracked.

HMD inertial sensors cannot always guarantee that the geometric transformations they represent are coherent. For



(a)



(b)

Fig. 1. These two sketches illustrate the problems introduced by drifting. After a sequence of head motions, inertial sensors accumulate errors and the orientation of the user head does not correspond any more to the avatar head's. Then, the user is in a very uncomfortable pose (a), and starts to turn his/her body unconsciously, until to feel comfortable again (b), going out of the field of view of the absolute sensor.

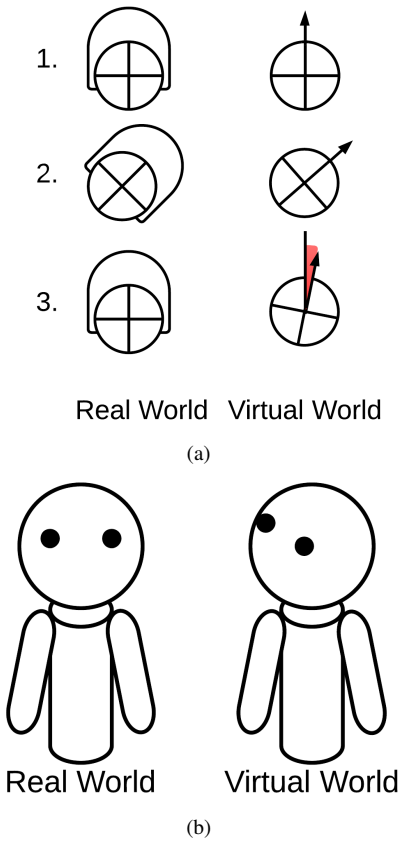


Fig. 2. Typical problems in VR systems when a full and complex tracking infrastructure is not being used. In (a), cumulative errors cause drift in the virtual environment. In (b), head body misalignment caused by poor body tracking system.

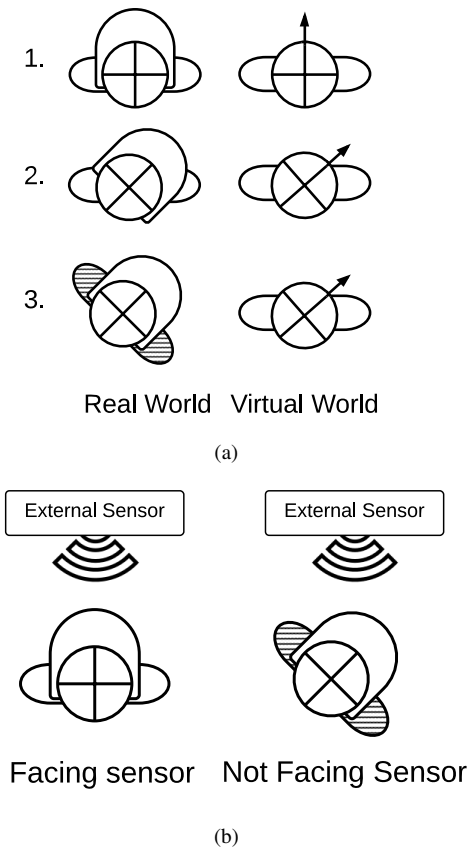


Fig. 3. Typical problems in VR systems when the user is allowed to move their feet. In (a), the avatar does not mimic the real user body orientation. In (b), turning the body may disturb the perfect functioning of an external sensor such as the MS Kinect.

example, a user facing a door in the virtual environment (VE) who is also facing a door in the real environment, after a sequence of head motions can be still facing the virtual door but not the real door any more. This happens because the inertial sensors accumulate errors as they cannot always distinguish accelerations and velocities caused by rotations from those caused by small displacements of the head. In other words, rotations of the user head in the real world are not necessarily registered with those of the virtual world, a phenomena we call "drifting", as depicted in Figures 14a and 2a.

Some VR systems also require the user body to be modeled for proprioception, i.e., for the user to be able to see their own body, increasing presence. In these systems, the body is driven to walk around the VE at the same time that the head rotations are tracked by the HMD. Most of these systems require a complex tracking system able to track the body and the head separately. If this tracking is not available or not accurate, another phenomena also occur which is derived from drifting: "head-body misalignment" (Figure 2b).

Many systems today try to take advantage of the MS Kinect device for body tracking. Although it is a non-expensive device widely available, it requires that the user remain standing at a certain distance in front of it. In this context, as the user is immersed with the HMD without any real world reference, some solution must be found to gently force the user *not* to turn the body away from the standard orientation and still avoid drifting. Naturally, such solution cannot rely on complex equipment as the Kinect is used exactly to make the hardware setup simple.

In Fig. 3, the problems of allowing free body rotations are illustrated. In Fig. 3(a), the user has its head forward in step 1 then turns it 45 degrees right in step 2. After some time in pose 2 it starts to get uncomfortable, therefore the user involuntarily rotates their body to a more relaxed pose in step 3. The avatar does not mimic the body rotation since we are only tracking the user's head, thus it creates again a head body misalignment. In Figs. 14b and 3(b), an external sensor such as the MS Kinect may not work properly if the user is not facing the device, therefore a solution to force the user forwards is needed.

In this paper, we introduce the *comfortable pose function*, an algorithm that iteratively computes the referential north without the help of any additional sensor, excepting the embedded HMD inertial sensors. We hypothesize that the algorithm is effective whenever the users do not move their feet and then present a validation experiment using a magnetometer. Experiments show how this algorithm can be used to solve the drifting problem. Moreover, we introduce a method using a hardware prototype to replace the HMD embedded sensors. Our prototype contains a magnetometer and a fusion function that combined with the comfortable pose function provides a

	Sensor Type	
Feet Rotation	*Incremental*	*Absolute*
Free	Not solved	Body orientation correction
Fixed	Drift correction	No correction needed

TABLE I
SYSTEM SCENARIOS AND POSSIBLE CORRECTIONS

reliable referential north even when the users move their feet. The data computed with this method are then used to gently force the user to stand appropriately oriented in front of the Kinect sensor. While our main contributions are in those two methods and their experimental analysis, another contribution is their application on a case study of a VR training simulator.

Table I categorizes the problems that can be solved using the comfortable pose function. In cases where an incremental sensor is used for head tracking, we can fix the drifting caused by relative errors only if it is assumed that the users have their feet fixed on ground. When we allow the users to move and rotate their body, the system must gently force their orientation in the real world towards an angle aligned to the real world by gradually changing the orientation of the avatar in the opposite direction. This is possible only if an additional absolute orientation sensor such as a magnetometer is available. The case when free user rotation is allowed in real world, and only incremental sensors are available is not covered here and is still an open research topic.

This paper is organized as follows. In Section II we review the literature to contextualize this work. Next, in Section III, we explain the illustration of Figure 4 describing how the comfort pose algorithm works and detail our software and hardware implementations in the context of VR. Section IV explains the experiments we developed to evaluate the proposed solution, and Section V brings a discussion and our conclusions.

II. RELATED WORK

Typical setups for body tracking in VR involve full tracking [1] [2]. Magnetic sensors as the Ascension Flock of Birds or optical sensors as the Vicon system have been used for more than a decade. The main drawback of these systems is their very high associated cost and complexity.

One of the main reasons for the delay in the widespread use of VR is the need for reducing the cost and complexity of the devices necessary to track the body and to provide immersive visualization. A number of previous works propose some kind of low cost techniques for tracking [3]. Despite the lower cost, however, many of those systems were still based on complex infrastructure to detect three-dimensional data.

More recently, the Kinect sensor gave a new impulse to the VR as it offers the possibility to track the body in 3D with a single device [4] [5]. Nevertheless, it requires the user to stand at a restricted area in front of the device. As the users tend to move a few steps and to turn the body inadvertently while immersed in the VE, Kinect is ineffective in many situations. In an attempt to minimize this problem, multiple Kinects have been used [6]. This offers an affordable solution

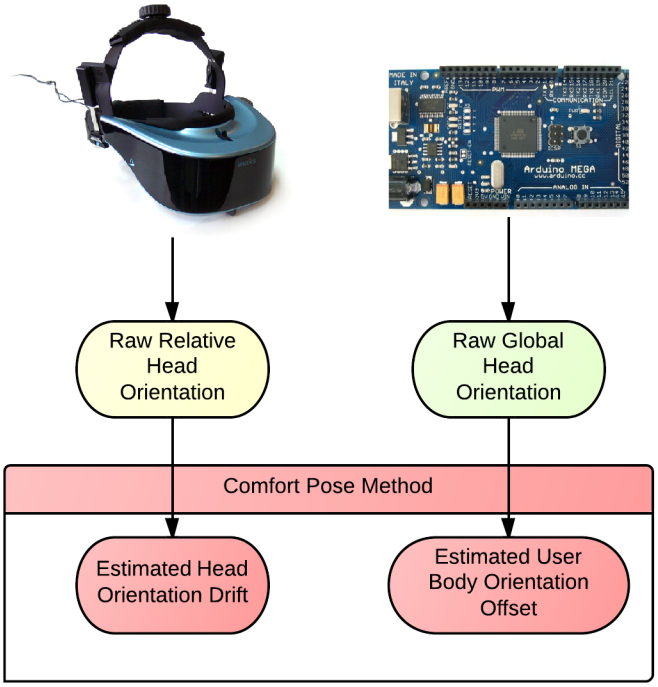


Fig. 4. Schematic representation of the role of the comfort pose function in the solution of two key problem in body tracking for VR. In the left using only the HMD embedded sensors, and in the right using an additional hardware.

in relation to a full tracking system. However, it brings us back to the situation where the tracking becomes complex, being necessary to synchronize the devices and process the input data according to the devices physical position.

Our approach to this problem, instead of multiplying the sensors along the environment, is to incorporate a sensor in the user's body and combine sensor data with algorithms to induce the user not to move away from the position of the stationary tracking sensor, the Kinect in our case.

Incorporating sensors is similar to what researchers do in robotics and augmented reality (AR). They place sensors on the robot and try to compute the robot's position based on them. In AR, cameras have been used with computer vision algorithms to detect features in the environment and estimate the camera position [7]. This is necessary because in several situations, both in AR and robotics, it is unfeasible to rely on fixed infrastructure for tracking. Still, some kind of tracking is necessary to align real and virtual objects in AR [8] and for the robot to know its position while navigating [9].

One interesting thing in our own approach is that we integrate data from a fixed device with data from incorporated sensors. Similar ideas have been tried before using complex installed sensor infrastructure plus wearable sensors [10]. In robotics, dead reckoning [11] is also very popular as a means for estimating tracking information for time periods when sensor information is not available between two instants when they are. One simple example is when a driver guided by a GPS receiver enters a tunnel. Using the car speed and the road map it is possible to estimate where in the tunnel the car is while GPS signal is not available.

III. The Comfortable Pose Approach

Here we describe our novel algorithm to find a referential north based on the assumption that humans spend most of the time with their heads facing forward in relation to their bodies, *i.e.*, they assume a comfortable pose. Later in this section we also describe the implementation and use of the algorithm, including the associated hardware. An experimental analysis and the application of this algorithm in VR systems is detailed in Section IV.

A. The Concept of the Comfortable Pose

The comfort pose algorithm is based on an estimative of the user's body orientation using only the head orientation. The head sensors provides a stream of orientations. The algorithm uses a window to analyze just the last seconds of data and infers the body orientation (see Figure 5). This can be done either on incremental or absolute sensor. The kind of information provided by this technique will depend of the scenario, given in table I. In the case of absolute sensors, the analysis will return an approximation of the user body rotation, for incremental sensors constraining the user body orientation, this analysis will be an approximation of the accumulated error. For incremental sensors and free user body orientation, nothing can be said.

Fig. 5. Idea behind the comfort pose algorithm.

A simple estimative of the body orientation is found calculating the median of the user's head orientation history. This provides an approximate body orientation for the given time frame. The inferred body orientation can then be used to solve the two problems discussed in the introduction.

B. Drift correction

Whenever inertial sensors are used, some drifting is expected that we aim to correct. The comfort pose algorithm has been implemented here to estimate and minimize the drifting of an HMD, as in Figure 6. Using the comfort pose method, we can estimate the user's body facing direction. If the users are not allowed to move their feet, it is expected that the body facing direction is always forward. If the orientation analysis returns a direction that is not aligned forward in relation to the body, then a drift has probably occurred in relative orientation sensors like those of an HMD.

The drift angle value is exactly the angle found by the history analysis, because the real user is not allowed to change their body direction. The drift can be compensated if the avatar's head in the virtual world is rotated in the opposite direction.

The avatar's head direction will then be fixed according to a fixing direction and speed. The fixing direction speed is given by the estimated offset. If, for instance, it is found that the offset is a few degrees left, fixing direction speed will be towards right, gradually incrementing the direction to the right and minimizing the drift.

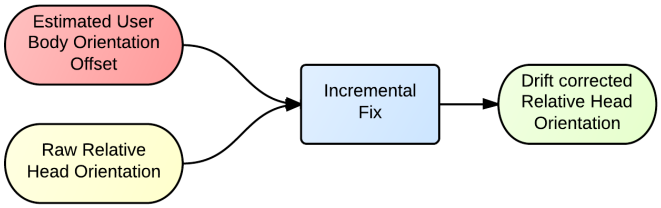


Fig. 6. Comfort pose used to correct HMD drift.

A Gaussian derivative function is used to define the speed behavior, as in Equation 1.

$$speed = offset \times Gauss(offset) \quad (1)$$

Constant values for the gaussian function are $a = 1/300$, $b = 0$, $c = 11$ degrees. This generates the curve in Figure 7. The value $1/300$ was chosen because of the average time step, and 11 because it generates a curve width that ends around 60 degrees, which is the field of view we achieve with the HMD. This curve augments the importance of smaller offsets, which will probably be a drift, and gradually lowers the importance of higher value offsets, to suppress an explosive behavior. Higher offset values can also mean that the user is doing a large voluntary head movement, thus the reduced importance.

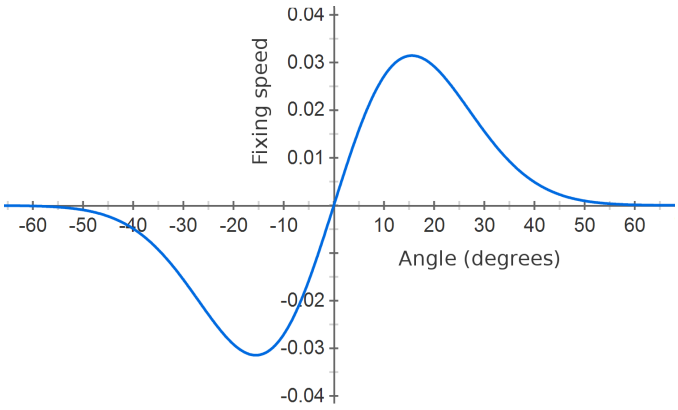


Fig. 7. Graph showing the fixing direction speed according to the estimated offset. Less importance is given to higher values.

The resulting drift correction value is updated with the offset fixing speed. The rotation read from the device is fixed with the drift correction value, and then stored in the rotation history to be used in the next time step in the estimated offset analysis. This whole process is summarized in the Algorithm 1.

Algorithm 1 Drifting correction

```
rotation = HMD.read();
offset = Median(history);
speed = offset × Gauss(offset);
driftCorrection.add(speed);
rotation = deviceRotation - driftCorrection;
history.push(rotation);
```

The drift correction value does not grow infinitely because the higher the fixing speed, the faster the rotation will return to origin. That way, the speed will shrink and will not cause an explosive growth of drift correction.

C. Hardware

We also designed and implemented a sensor hardware prototype based on the Arduino platform to perform global tracking of the real user's head orientation. The prototype contains a 9 degree of freedom sensor consisting of 3 axial accelerometer, gyroscope and compass. The compass, which is not present in the average HMDs, is the sensor that allows us to perform a correct global tracking of the user head orientation without drift.

Since the Arduino features three different kind of sensors, a sensor fusion technique had to be applied to obtain a consistent orientation. We adapted an existing attitude and heading reference system Arduino sketch to perform the readings [12].

We then fixed the Arduino on top of the HMD in such a way that they suffer the same movements (see Figure 8). This allows us to compare the readings. This setup also permits us to override the readings of the HMD with the readings from the Arduino, in order to have correct orientation reading. This hardware is used in an experiment described later in this paper to globally track the real user head orientation, which allows us to validate the drift correction software presented in Section III-B.

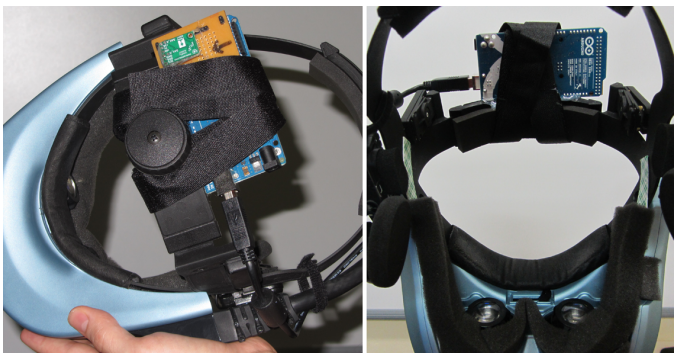


Fig. 8. Assembly of the HMD with the Arduino board and the magnetometer.

D. User body orientation correction

In practical situations, users do not feel comfortable in having their feet fixed. Then, we propose to use the same methods and hardware described above in a different configuration to give the user total freedom but taking measures to enforce the *stand in front of the Kinect* condition. Again, the hardware prototype of Section III-C is rigidly placed on the top of the HMD. The HMD sensors are now disabled as they are overridden by those of the hardware prototype. The user view is then updated based on our sensor hardware, including magnetometer readings.

As said before, the magnetometer can provide a grounded reference north. However, as it is placed on the user's head, it will give the orientation of the head instead of the body. The body orientation must then be estimated. We estimate the user's body orientation using the same comfort pose algorithm described above. In simpler words, we assume that if the user's head remains a long time facing a given direction, his/her body is probably turned to that direction as well.

Knowing the estimated body orientation, if it deviates from the center, small rotations are gradually applied on the user's avatar to the same direction. This will force the user to look, by turning the head, in the opposite direction to keep seeing the same scene. This eventually induces the users to move their feet, rotating the body to be in a comfortable pose aligned with the head. This rotation will bring the user back to the center, i.e., facing the Kinect sensor. We used this approach on a case study with a VR simulator as described in Section IV-D. Our observations show that the users tend to remain in the suitable orientation in front of the Kinect, which was not the case in previous experiments where the correction was not applied. The hypothesis H2 is then also confirmed.

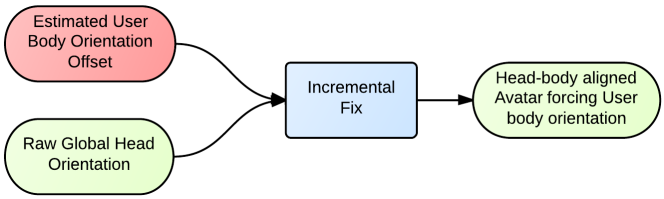


Fig. 9. Comfort pose used to force user body facing direction.

Since we can infer the body orientation using only the head tracking information, if the user is misaligned from center, we can apply the same rotation in the virtual world, forcing the user to look in the opposite direction. After some time, the user will eventually move their whole body to the comfort pose, realigning it towards the system.

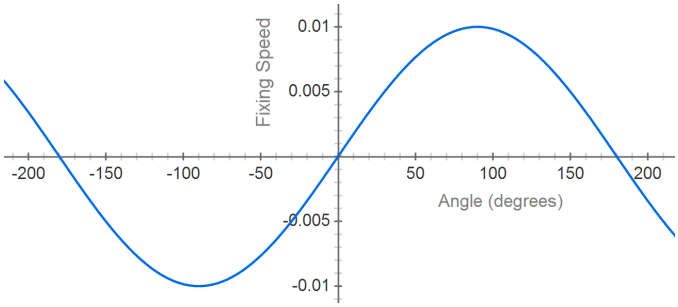


Fig. 10. Graph showing the fixing speed and direction for fixing the user forwards.

The algorithm to force the user forwards differs from the drift correction only on the speed function and the applied rotation direction. We started using a similar derivative Gaussian function to calculate the speed correction, but a sin curve multiplied by a small constant was used instead, as it is similar to the Gaussian derivative curve and covers a higher angle range, as seen in Fig. 10. Consequently, the correction works even if the user is almost 180 degrees rotated. The correction will be applied on the virtual world avatar's body, instead of the head, and to the opposite direction. The full algorithm can be seen in Algorithm 2.

Algorithm 2 User body orientation correction

```
rotation = arduino.read();
offset = Median(history);
speed = Sin(offset) × c;
avatarBodyRotation.add(speed);
history.push(rotation);
```

In this algorithm, c has a constant value of *0.01*. It is important to notice that, contrarily to the head direction, the avatar body rotation is maintained in each iteration. Therefore, we do not need to incrementally maintain the error as in the drift correction, so we only add the speed directly to the avatar.

IV. Experimental Evaluation

A. Hypotheses

We consider the use of our implementations described in Section III to:

- enforce the alignment of the head and body of a user's avatar
- enforce alignment of the user body with the real environment

We then elaborate two hypotheses, as described below:

- ***H1.*** The comfortable pose algorithm provides drifting correction as accurate as the hardware implemented magnetometer measurements.
- ***H2.*** The comfortable pose algorithm combined with the hardware implemented magnetometer prototype is able to induce the user to remain facing toward the same direction of the real space.

Later, we propose experiments to test each hypothesis, which we describe below.

B. Evaluating drifting and head-body misalignment

The hardware prototype of Section III-C is rigidly placed on the top of the HMD (as seen in Figure 8). This allows us to obtain three simultaneous measurements for the head angles: HMD-based, magnetometer-based, and software-based. We know that the HMD is susceptible to drifting and that the hardware implemented magnetometer measurements are always correct. Then we assume the magnetometer-based measurements as a baseline and compare these data with the data computed from the other measurements. This allows us to measure how much the HMD data and the software data differ from the correct orientation, thus testing hypothesis H1.

Instead of testing with users, we decided to stress the system to the maximum and to make controlled movements that are reproducible. Users would rather make random movements that are too complex to analyse. Then we manually applied a sequence of movements that repeatedly accelerates and decelerates the device in various directions. The sequence we used lasts for about 2 minutes and is given below:

1) 5 times turn alternately from right to left and vice versa
2) turn 45° up and apply the 5 times turn alternately from right to left and vice versa
3) turn 45° down and apply the 5 times turn alternately from right to left and vice versa
4) translate 5 times without turning following an ∞ symbol in the vertical plane

Figure 11 shows the measurements obtained from the three different tracking methods for six repetitions of the experiment. Notice that even with movements made manually, the magnetometer based readings are almost identical, with only barely noticeable differences in time. Also, notice that the active user motion follows the same pattern for all three tracking modalities. However, while the HMD data diverges from the actual measurements of the magnetometer, the corrected data obtained with our comfort pose function follows the actual data much closer, ending at the same position.

From the data obtained we verify that the algorithm is an adequate software solution for the drifting problem depicted in Figure 2a that does not require any additional hardware, thus proving hypothesis H1. This method is applicable in VR setups as simple as those containing an HMD and no other tracking sensor or infrastructure. In the case a sensor like Kinect is used to track body motion and assuming that the user does not move their feet, this solution guarantees that the virtual head will always be aligned with the real head, avoiding the problem depicted in Figure 2b.

C. Evaluating deviation from the real world

With globally correct user head orientation readings, it is also possible to solve another spectrum of problem, like forcing the user to always stay oriented forward, as shown in Figure 9. This is helpful when using other interaction devices that require the user to keep facing the same direction, like the Microsoft Kinect, for instance.

We then proposed an experiment where users are invited to move around a virtual environment and select objects. There are two system setups. With the first setup the users received the correctional rotations described above. With the second setup the users use the system as is, without correctional rotations. A within-subjects experiment design was used to maximize the number of collected samples. This means that all users tested the two setups. We swapped the order in which the setups were presented to each user to avoid bias. We hypothesize that the users will perform better (faster and more accurate) in the setup with corrections as they will have consistent real-virtual positions all the time. Althought this might be thought as obvious, the hypothesis can only be proven if the approach is correct, the implementation is sound, and the imposed constraints do not make the users feel they are not in control of the interaction.

All users were then asked to move in the environment, and point and select all the fire estinguishers they can find. To keep the users focused, a path indicated by arrows is drawn on the ground in the VE. The fire estinguisher has been choosen as it is easily spotted by their shape and color. The users are not informed about the purpose of the experiment. Therefore, they do not know that there is a pose correction or that we might be interested by that question.

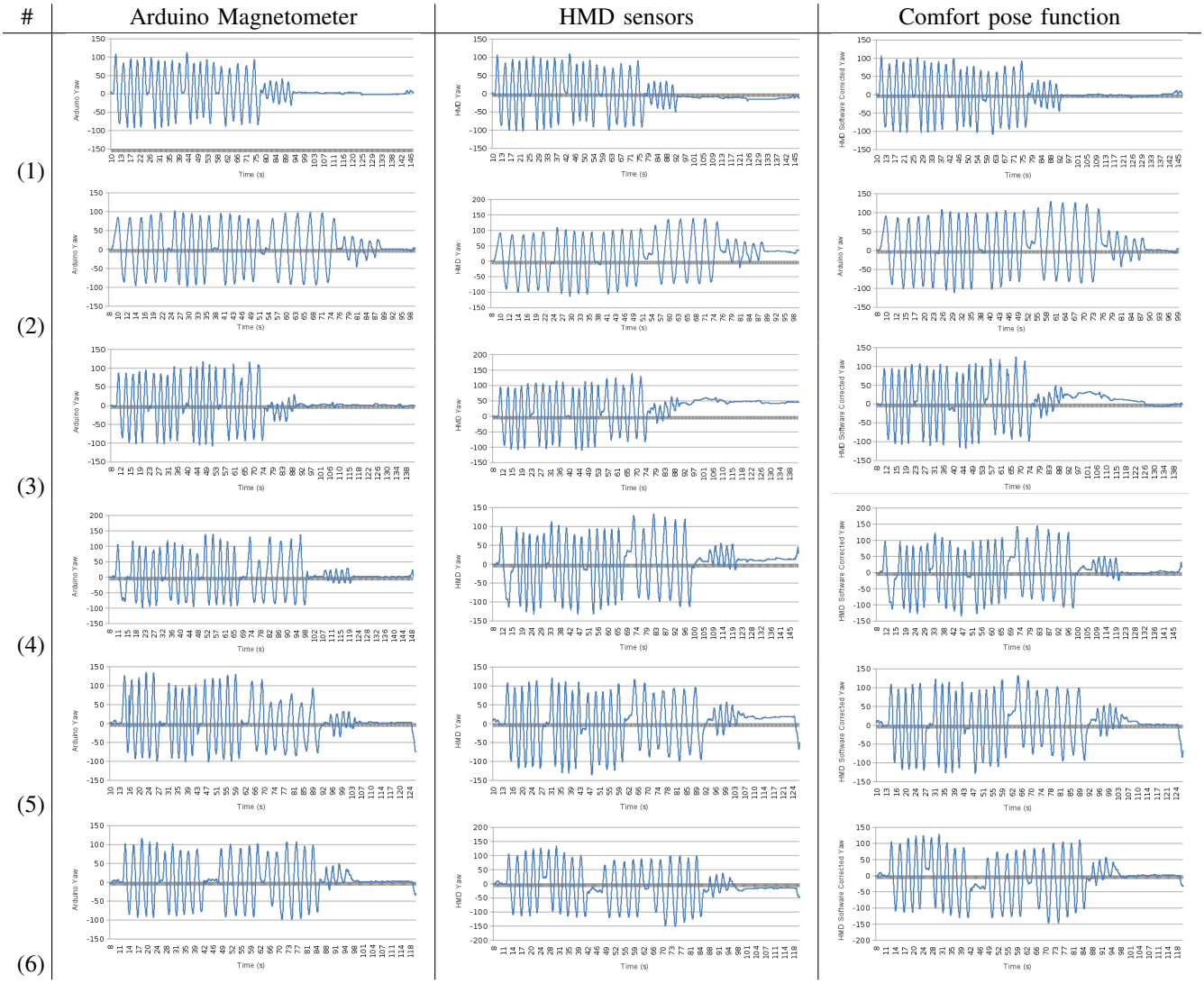


Fig. 11. Time normalized examples of data obtained from 6 repetitions of the experiment. The first column shows the baseline magnetometer reading, the second column shows the raw data measured from the HMD sensors, and the last one shows the data corrected with our comfort pose function. Yaw values are angles in degrees.

The users were also invited to fill a characterization questionnaire before entering the experiment and an opinion questionnaire just after they finished the test.

1) Population: The population of the experiment consisted of five randomly selected users. The users tested the system twice, once with the body rotation correction and another without it. Three users started the experiment first with the correction and 2 users started the experiment without the correction first to avoid bias.

The experience of the users using VR systems ranged from no experience to experienced. There was 1 female subject and 4 male subjects. 2 Subjects had previously used the MS Kinect controller.

2) Results and Discussion: We keep track of the yaw component of a magnetometer attached on the body of the subject, that indicates how much the subject rotated their body. When the rotation correction was active, it was possible to observe that the users keep their body directed forward.

The raw data is presented in figures 12 and 13. This data will be further explored in the future.

During the experiment the users who performed the task with the correction pose algorithm remained aligned to the Kinect whereas the users who performed the task without the algorithm showed a misalignment. Although it was possible to perceive it in the experiments, we still need to process the captured data in order to provide a better insight.

D. Case study: a VR simulator

With the confirmation of the main hypotheses we felt encouraged to apply the methods presented in this paper on a simulator developed in our lab, the AES-risk [13] [14]. This simulator allows a user to walk in a VE that represents the facilities of a power distribution company. There, the user is invited to explore the environment and select any objects they think might be a potential risk (see Figure 15). The simulator is used to assess the risk perception competences of the company's employees aiming at improving safety.

This simulator is based on the architecture depicted in

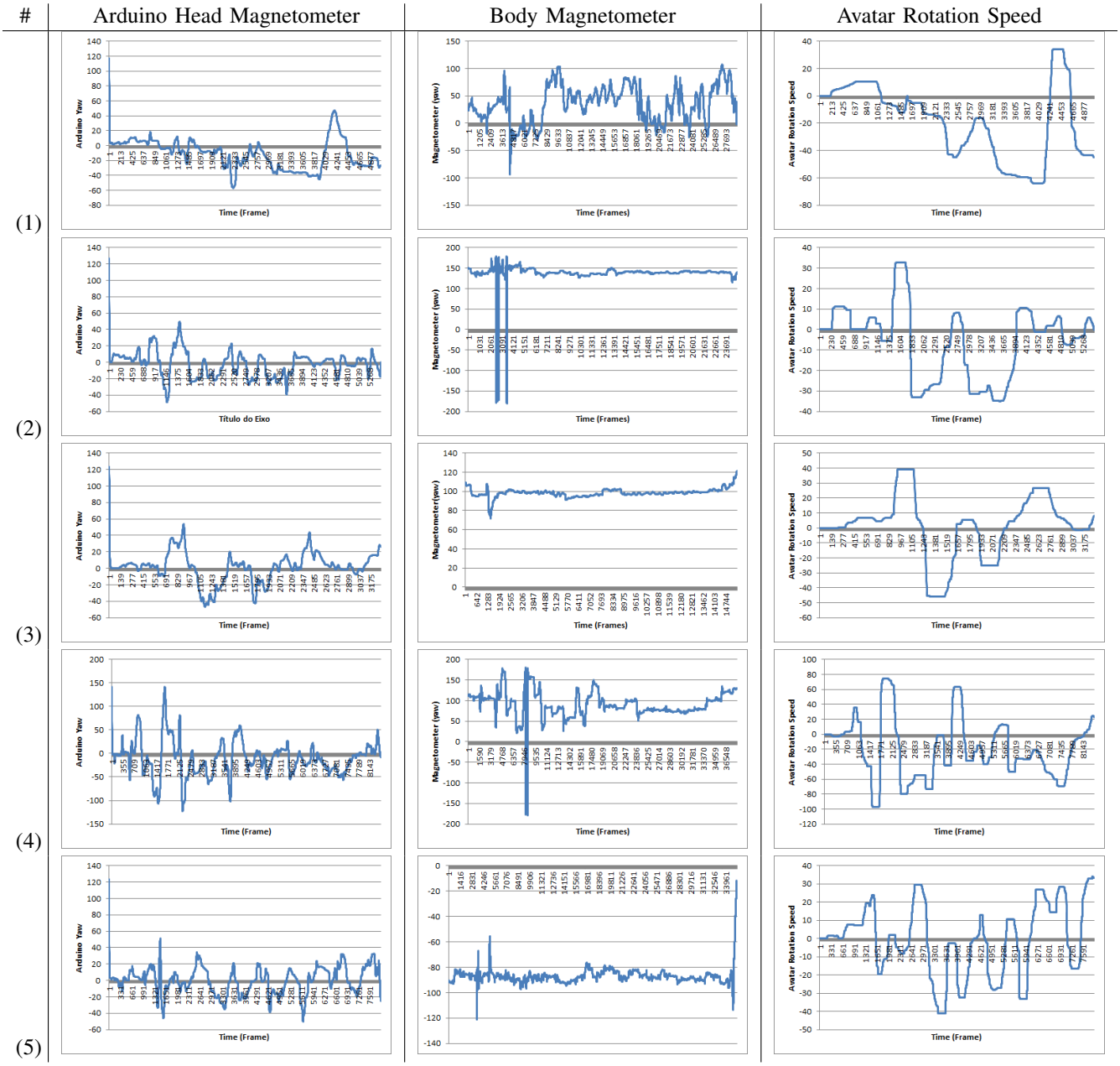


Fig. 12. Raw data of the sensors and avatar rotation correction speed. The first column shows the head magnetometer reading, the second column shows the raw data measured from the body Magnetometer sensors, the third column is the avatar rotation correction speed. Yaw values are angles in degrees.

Figure 16. The user interacts with the system using three different devices: the HMD, the Kinect and the gamepad. The system is based on the Unreal Development Kit (UDK) game engine. The integration of UDK with Microsoft Kinect is made through the OpenNI Unreal Implementation (NIUI). The use of Kinect is strategic to track the user's limbs and apply their motion to the user's avatar.

During the system development many users seemed to have enjoyed the experience of having control of avatar limbs using body motion. This actually increases the feeling of presence. Besides, we also use a standard gamepad, which is readily supported by UDK and has a very accurate response to trigger actions. The gamepad was used to provide constant speed locomotion while the push of any of the gamepad buttons select or deselect items. The HMD maps the head movement to quaternion rotations. We use a dll bound to UDK to pass this information into the engine. We handle the quaternion data inside UDK, converting them to rotators for compatibility.

One important element in the design of VR simulators is to ensure that the user task is not significantly influenced by the user interface itself. We have then made experiments to assess different interaction techniques with more than 40 users [15]. When comparing them, however, we noticed that drifting and user's body misalignments were injecting too much error in the measurements. This happened because each user eventually assumed very different poses to complete the same task. As some poses were tiresome, some of them presented very low performances comparing with the average, resulting in

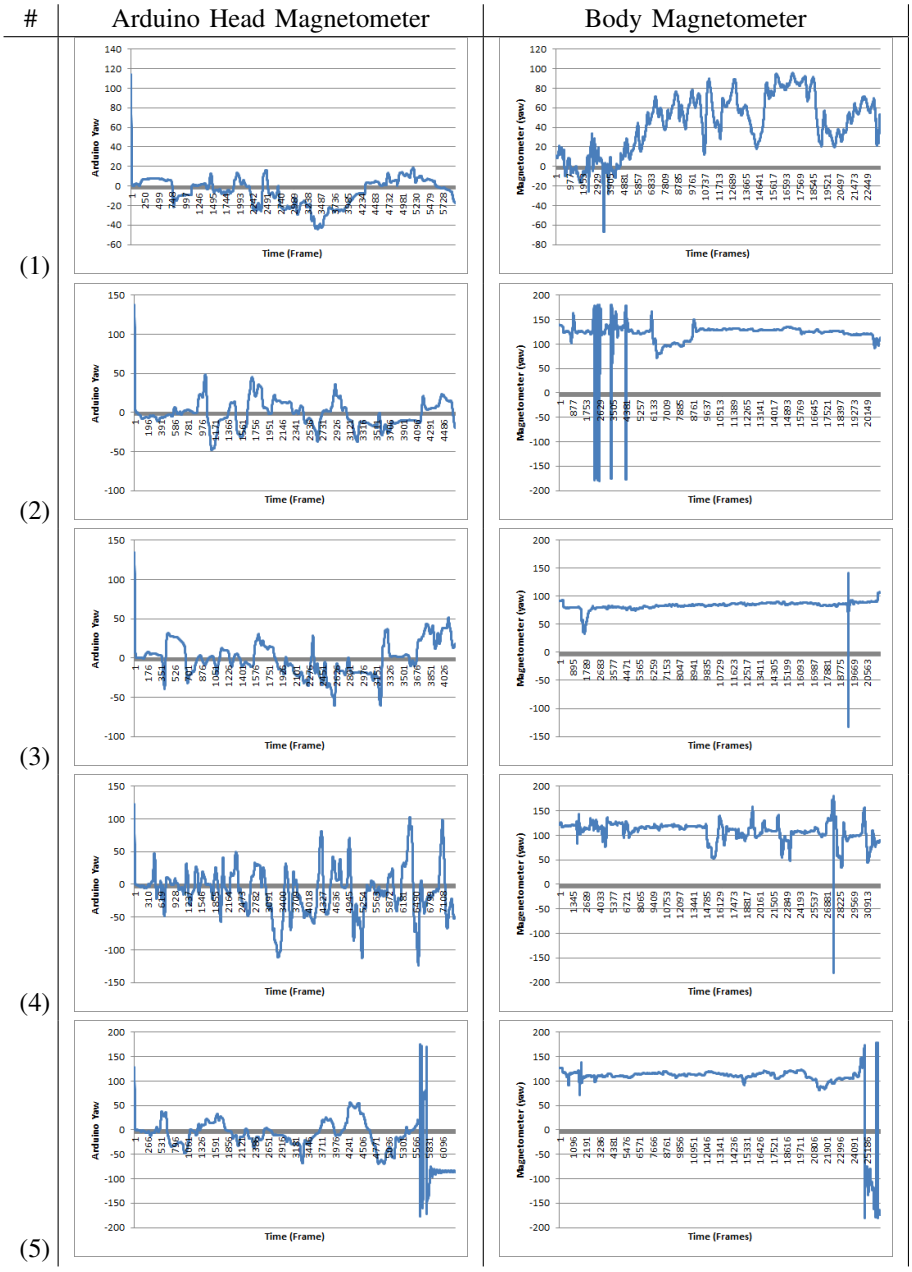


Fig. 13. Raw data of the sensors when there is no correction. The first column shows the head magnetometer reading, the second column shows the raw data measured from the body Magnetometer sensors. Yaw values are angles in degrees.

less statistically significant data. Only after the application of the algorithms and methods described here we could obtain consistent data.

V. Discussion and Conclusions

In this paper we presented a unified software solution for two common tracking problems in VR systems. Both problems exist whenever a full tracking infrastructure is not available, which is often the case in current systems based on the Kinect device and inertial sensors. Even if the Kinect can be barely used as a position tracker, it was not conceived for this task and is not sufficiently precise for most VE applications. Our solution for this problem is based on the assumption that users are more comfortable when looking forward and, thus, privilege this pose. Our main contribution is in a comfort pose function that decides what is forward based only on inertial sensors mounted on the head. Thus, our solution remains independent of Kinect or any other tracking device.

We demonstrated that the function can be applied to correct the drifting caused by HMD sensors without additional hardware. We used a magnetometer in an experiment to evaluate the function performance. In our experiments we noticed that the magnetometer may be influenced by magnetic interference from the HMD, which we solved by placing the magnetometer away from metallic parts. We focused our attention in the yaw movement, which is the one the accelerometers have more difficulty in detecting accurately. We used yaw and pitch movements but avoided roll. We did so because roll is much

(a) (b)

Fig. 14. Two screenshots of our VR simulator for risk perception analysis. Notice that the user can see her/his own body, which increases presence.

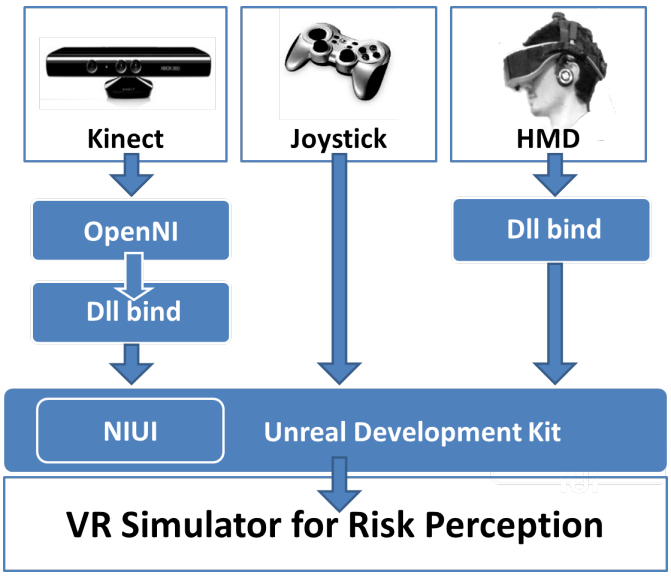


Fig. 15. Overview of the VR simulator architecture. User can interact with three devices simultaneously. The dll binds are a requirement of UDK engine to include hardware devices that it does not support by default.

affected by sensor axes misalignments, i.e., the axes of our hardware prototype should be perfectly aligned with the HMD axes to avoid perturbations.

We presented samples of data collected in the experiments. While our first idea was to repeat the experiment at least 30 times and apply a more statistically valid analysis, we dropped this idea because the different hardware provide different latencies and sensitivity that are not comparable. Notice, however, that the monotonous data for the 6 samples provided in Figure 11 indicate that they are reliable.

We also demonstrated that the comfort pose function can be used to gently force a user to keep his/her body aligned with a motion sensor fixed in the real world. While this is very useful, as we have observed with our risk perception VR simulator, we acknowledge that different applications may stress the tracking approach differently and should be further evaluated.

We then suggest as a future work that formal comparative experiments with a larger number of users focus on this issue. A second magnetometer placed on the user trunk or waist, should be necessary to quantitatively compare subject groups with the correction enabled and disabled. We also identify research ground for doing both correction techniques, user body and drift, using incremental sensors while not fixing the user feet. Finally, another path to be followed in future works is to make longer experimental sessions as it is common that inertial sensors present cumulative errors that highly increase over time. We believe that our method's results will be even more interesting in longer sessions.

Acknowledgment

We would like to thank Martin Reus form valuable help in the hardware implementation, as well as all the users who kindly tested the system. We also thank AES Sul for funding this project, and CNPq-Brazil for the financial support through projects 311547/2011-7, 485820/2012-9, 302679/2009-0 and 305071/2012-2.

References

[1] F. Kellner, B. Bolte, G. Bruder, U. Rautenberg, F. Steinicke, M. Lappe, and R. Koch, "Geometric calibration of head-mounted displays and its effects on distance estimation," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 4, pp. 589–596, April 2012.

[2] J. J. H. Carlo Camporesi, Marcelo Kalmann, "A framework for immersive vr and full-body avatar interaction," in *Virtual Reality Conference (VR), 2013 IEEE*, March, 2013, pp. –.

[3] S.-M. Rhee, R. Ziegler, J. Park, M. Naef, M. Gross, and M.-H. Kim, "Low-cost telepresence for collaborative virtual environments," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 1, pp. 156–166, Jan.-Feb. 2007.

[4] B. Lange, A. Rizzo, C.-Y. Chang, E. Suma, and M. Bolas, "Markerless full body tracking: Depth-sensing technology within virtual environments," in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011*, Orlando, FL, 2011.

[5] T. Motta and L. Nedel, "Deviceless gestural interaction for public displays," in *Proceedings of the 2013 XV Symposium on Virtual Reality*, ser. SVR '13. Washington, DC, USA: IEEE Computer Society, 2013.

[6] S. Satyavolu, G. Bruder, P. Willemsen, and F. Steinicke, "Analysis of ir-based virtual reality tracking using multiple kinects," in *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, March 2012, pp. 149–150.

[7] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, ser. IWAR '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 85–. [Online]. Available: http://dl.acm.org/citation.cfm?id=857202.858134

[8] S. You, U. Neumann, and R. Azuma, "Hybrid inertial and vision tracking for augmented reality registration," *Virtual Reality Conference, IEEE*, vol. 0, p. 260, 1999.

[9] J. Borenstein, H. R. Everett, and L. Feng, *Where am I? Sensors and Methods for Autonomous Mobile Robot Positioning*. University of Michigan, 1996. [Online]. Available: http://books.google.com.br/books?id=QYBD7vHWUgIC

[10] M. Akula, S. Dong, V. R. Kamat, L. Ojeda, A. Borrell, and J. Borenstein, "Integration of infrastructure based positioning systems and inertial navigation for ubiquitous context-aware engineering applications," *Advanced Engineering Informatics*, vol. 25, no. 4, pp. 640–655, 2011.

[11] S.-h. Won, W. Melek, and F. Golnaraghi, "A fastened bolt tracking system for a hand-held tool using an inertial measurement unit and a triaxial magnetometer," in *Industrial Electronics, 2009. IECON '09. 35th Annual Conference of IEEE*, Nov., pp. 2703–2708.

[12] P. R. . Electronics. (2012) Pololu minimu-9 + arduino ahrs (attitude and heading reference system). [Online]. Available: https://github.com/pololu/MinIMU-9-Arduino-AHRS

[13] V. A. Jorge, A. Hoppe, A. Maciel, L. Nedel, G. Reinaldo, F. Faria, J. Oliveira, and P. Montani, "Development of an immersive vr simulator using the unreal development kit," in *Proceedings of the 2012 XV Symposium on Virtual Reality*, ser. SVR '12. Brazilian Computing Society, 2012.

[14] V. Jorge, L. Nedel, A. Maciel, J. Oliveira, and F. Faria, "Aes-risk: An environment for simulation of risk perception," in *Virtual Reality Conference (VR), 2013 IEEE*, ser. Research Demos, March, 2013. [Online]. Available: http://ieeevr.org/2013/program/activities/researchdemos

[15] V. Jorge, A. Maciel, L. Nedel, J. Oliveira, and F. Faria, "What is the effect of interface complexity on risk perception tasks?" in *Virtual Reality Short Papers and Posters (VRW), 2013 IEEE*, ser. Posters, March, 2013. [Online]. Available: http://ieeevr.org/2013/posters