The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

A Simple Inexpensive Interface for Robots using the Nintendo Wii Controller

Sven Olufs and Markus Vincze

Abstract— **To have a robot at home might be great fun: it could fetch and carry things. However it remains open how to teach the robot the places it should go to in a manner that is cheap and entertaining for the user. This paper presents an easy-to-use interface that takes the robot on a virtual leash: using the Nintendo Wii remote the user can go towards target places while pointing at the robot. Using the inbuilt infrared camera and accelerometers and a couple of LEDs on the robot, the robot will follow the user. We show how a particle filter and an interacting multiple model (IMM) Kalman can be configured such that simple hand gestures with the Wii make the robot follow the user's intention. The concept has been implemented on a mobile robot developed within the robotshome project. The robot leash interface has been tested with 12 volunteers who are interested in new technology but have never controlled a robot. The result is that most users could within a few minutes show the robot the first three places in a home environment. Given the little cost of the interface (about $ 50) the proposed robot leash is a promising human robot interface.**

Disclaimer: *We do not propose the usage of the Nintento Wii remote for commercial robot projects. It is only used for proof of concept.*

I. Introduction

Apple released the first personal computer "PC" on June 5th, 1977 for the mainstream market. Experts proposed that a commercial success of personal computers will depend on their market price and usability. Apple's computer was cheap ($1298), ease to use via onboard BASIC, and thus became the first successful home computer for the mainstream market. Today we know one of the main reasons for the success that was totally underestimated: the Apple II was fun to use. Due to a wide and quickly growing developer community in the pioneering age of personal computers, a wide range of (free) computer games was available. Even people without any previous interest in computers found easy access to this "new world" via games. Today we see the same development with Appls iphone. That's why we propose that an interface to a robot *must* be fun for the user, and inexpensive.

The BASIC interface of the Apple II was quite primitive compared to today's computer/robot state of the art interfaces, there was no graphical interface or no mouse. But to assume that current user interfaces are much easier to use is a post hoc fallacy. Let's face it: there are no things in the real world like "Windows", "Start" menus or "Sliders" as in Microsoft's Windows or Apple's MacOS. Even the peripheral interfaces itself, i.e., mouse, keyboard etc., is for the most people still a myth, especially the elderly. In the case of a robot interface this is even more difficult. For instance, using a joystick to steer the robot like a toy RC-car is easy, but hard to manage for inexperienced users. In fact, even for many simple interfaces an assistance system would be needed.

In this paper we present an inexpensive interface (total hardware costs less than $50) for mobile robots that is easy-to-use and fun-to-use. Instead of considering the robot as a mere piece of technology, we consider the robot as a pet. For instance, we control the robot like a dog, i.e. by a virtual leash to guide it and by using simple gestures. Commands for a robot can be "stay", "follow", "bring xy" etc. In practice the recognition of spoken words or the visual detection of the user & gestures is quite hard. We equip the user with dedicated hardware that simplifies the recognition process: the Nintendo Wii Remote.

The Wii remote (approx $40) is a quite interesting device for human robot interfaces, because it is inexpensive and equipped with an infrared camera. The main idea is to equip the robot with patterns of infrared beacons (custom build approx $10) that are used to calculate the relative pose to the user. An additional sensor in the Wii allows also tracking the pose when no beacons are visible. If the robot has a basic understanding of the environment and its pose, we can also extend the interface with "pointing to things" for manipulation.

This paper is organised as follows: The next Section gives a brief overview of the related work. This is followed by a description of the concept of the interface. In section 4 the Nintendo Wii is presented. Sections 5 and 6 present the technical details of our implementation as well as the experimental setup. Experimental results are presented in Section 7. Finally, we discuss the results and the approach in section 8.

II. Related Work

Our interface is based on the idea that for interaction we identify the position of the user and the direction he/she is pointing. The related literature propose many approaches, for example vision based ones: Waldherr et al. [1] proposed an vision based gesture interface that is able to detect and recognize up to four gestures (Left, Right, Turn, Stop, etc). It is based on colour tracking for human identification and a neuronal network for gesture detection. The user must stand face to face with the robot. The direction of pointing is indirectly obtained using the gestures. Instead of neuronal networks, Isard and Blake [2] used active contours to track

The research leading to these results has received funding from the European Community's Sixth Framework Programme (FP6/2003-2006) under grant agreement no FP6-2006-IST-6-045350 (robotshome)

Sven Olufs and Markus Vincze are with the Vienna University of Technology, Automation and Control Institute, Gusshausstrasse 30 / E376, A-1040 Vienna, Austria

hand gestures and the human silhouette with a multi hypothesis system in camera images. In contrast to [1] it is able to detect the orientation of the fingertips, but not to detect the arm of the user. [3] is using an opportunistic approach by attaching the camera directly to the human head instead of the robot. The user's hand is tracked through colour trackers and gestures are detected through Hidden Markov Models. The environment is mapped in 3D, the pose of the human is known.

Another approach is the usage of external devices. The easiest device are buttons or touch screens. In this case, the pose of the user is not important. Recent robot projects like Minerva [4] have shown that such interfaces are suboptimal: Due to their technical nature they were not understood by most of the audiences. Kemp et al. [5] used a laser pointer to point to things that the robot shall grasp. The robot is equipped with an omnidirectional camera with an appropriate filter for the wavelength of laser i.e. only the laser beam is visible in the image. An additional stereo camera detects grasping points on the objects. Corrales [6] used an opportunistic approach like [3]: the user wears a jump suit that is equipped with gyroscopes and accelerometers. The position of the user is known through an external localisation system.

The use of the Wii as robot interface has become quite popular in the last years. Rehm et al. [7] uses the accelerometers for gesture recognition, while Connolly [8] used the accelerometers for robot arm control. Guo and Sharlin [9] used two Wii remotes to control humanoid robots with the accelerometers.

Please note that the use of IR-Cameras and beacon is quite popular [10] in the Augmented Reality Community. The method is vice-versa to our approach: Multiple external ir-cameras are used and the user is equipped with beacons. The matching of the beacons is commonly done with the well known 3 point or 4 point algorithm.

III. Concept of the interface

The main idea of the interface is to guide the robot through the environment on a leash like a dog, i.e. to show it around in an unknown environment: the robot is guided to a position through straining at the leach. The use of the Wii Interface and robot is easy: The user takes a position in front of the robot and points the Wii to the centre of the robot. The relative pose of the user is detected using the infrared beacons on the robot and the Wii's infrared camera. The robot tries to keep a constant distance (1.2m) to the Wii by moving forward or backwards. For instance, a distance of 0.8m will result in a backward motion of the robot (pushing). The robot also tries to keep the angular orientation to Wii. That is, the robot will turn left when the user moves the Wii to the left. This allows the user to guide the robot through the environment by walking "in front" of the robot. Once the environment is learned (or a-priori known) we can assign nodes in the environment and send the robot to the nodes through gestures. The nodes are also learned through gestures. Using nodes simplifies the use of gestures due to the fact that the user does not have to do "precise" pointing all time. Now we use can use another application of the interface i.e. "point to things" that the robot shall manipulate. This is possible because the robot knows its position in the environment and can detect and track the user by means of the Wii remote. Please note that manipulating does not always mean to grasp objects: For instance, if the user points the remote at a lamp, it can be switched on/off, or pointing at tv set can be used to change the channel. To enable this, the robot would dispose of a device to remotely control appropriately euipped power plugs or be integrated into a home automation system.

IV. Nintendo Wii Remote

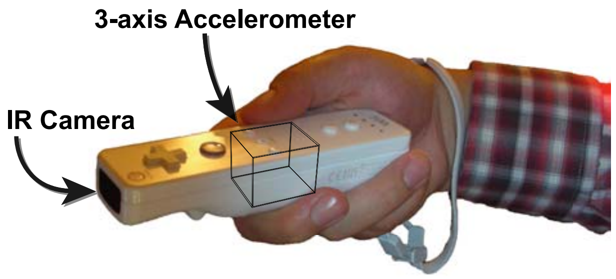


Fig. 1. Nintendo Wii Remote

The Nintendo Wii remote is a commercial product that is used as wireless bluetooth input device or "game pad" for the Nintendo Wii video game console. The Wii is the fifth home video game console released by Nintendo. The console is the direct successor of the Nintendo GameCube. Figure 1 depicts the Wii Remote. It has been available since spring 2007 worldwide and it is inexpensive. It assumes a one-handed remote control-based design instead of the traditional gamepad controllers. The remote has extra sensors for measuring the relative movement and rotation of the controller in 3D Space via 3-axis accelerometers. The sensor is located in the centre of the remote. This sensor enables us to recognize gestures, e.g. using hidden Markov models [11], [7]. Nintendo and AiLive Inc. offer a gestures recognition software development kit that uses SVN and neuronal networks. The remote also features a simple actuator: the "rumble kit" for tactile feedback. Furthermore, it contains a CMOS "PixArt Multi-Object Tracking" infrared camera that is able to simultaneously track up to four infrared beacons. Nintendo uses two active infrared beacons to initialise the 3D pose of the controller in a 2D Plane (the beacons are usually placed below the screen connected to the Wii game console). The sensor has a resolution of 80x60 pixels and is able to detect blobs at this resolution (only bounding boxes of the blobs are provided through the interface). Additional information i.e. centre of gravity of the blob is provided with $\frac{1}{4}$ sub pixel precision.

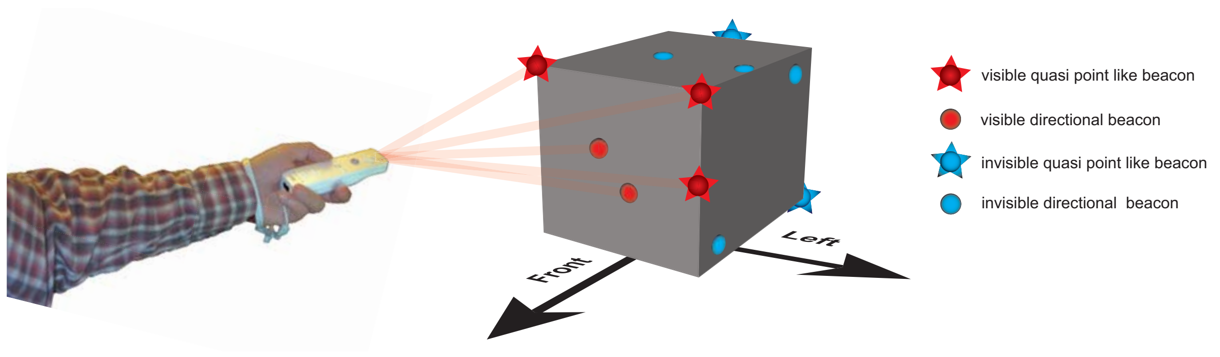

visible quasi point like beacon
visible directional beacon
invisible quasi point like beacon
invisible directional beacon
Front
Left

Fig. 2. Technical implementation of our Approach: The user points the remote at the robot (gray box) to localize the user relative to the robot. The a-priori known active beacons are detected through the IR-camera of the remote and the pose is tracked using monte carlo techniques. Each side of the robot shows a unique beacon pattern that is easy to distinguish. Two kind of infrared beacons are used: directional light source beacon and quasi point like light source beacon with a angle of beam of 30 degree and 180 degree respectively. The approach takes also the visibility (occlusion by the robot) of the beacons into account.

Nintendo announced in 2008 an extension for the Wii Remote: the Motion-Plus. The device incorporates a dual-axis "tuning fork" angular rate sensor, which can determine rotational motion. The information captured by the angular rate sensor can then be used to distinguish true linear motion from the accelerometer readings. This allows for the capture of more complex movements than possible with the Wii Remote alone.

V. Technical Implementation

Our technical implementation is inspired by the Nintendo Wii video game console: It is based on the concept that the user actually points to the desired position on the screen with the input device itself instead of controlling the console via a joystick or mouse. This is done using pose estimation via triangulation using two active a-priori known beacons. We use a different setup to obtain the position of the controller in 3D by extending the beacon setup itself to 3D. We equip the robot with a-priori known beacons with specific unique pattern on each side of the robot that is easy to distinguish (see Figure 2). We use two kind of infrared beacons: directional light source beacon and quasi point like light source beacon with a angle of beam of 30 degree and 180 degree respectively. Both types of beacons are made of five infrared diodes that are aligned in quasi star like orientation. The quasi point like beacons uses an additional prism to extend the angle of beam. Please note that a typical infrared diode has a angle of beam approx. 16 degree.

The pose is estimated by applying a variant of Markov chain monte carlo filters to the data: the monte carlo localisation (MCL). MCL is a popular approach to self localisation of mobile robots introduced by *Fox et al.* [12]. It is a Bayesian probabilistic method, in which the position of the robot is represented by a set of n weighted particles. Each particle contain a "believed" position with an assigned probability π. The pose of the robot is estimated by using the observations as a likelihood function of the believed poses/states while MCL attempts to maximise the likelihood of the beliefs. We denote $\Phi(x,y,z,\alpha,\beta,\theta)$ for the pose of the Wii (and particles): $x/y/z$ represents the absolute position in polar coordinates and α,β,θ for roll, pitch and yawn respectively. The point $(0,0,0)$ denotes the centre of the robot.
MCL uses the sensor readings of the infrared camera, accelerometers readings and the a-priori known visibility information of each beacons to estimate the pose of the Wii. It is assumed that even a point-like beacon is not visible through the robot body itself. The MCL framework provides a framework that is able to deal with outliers and provides also a framework for tracking using sensor and motion models i.e. even if no sensor data is available. The Wii remote is able to track and detect up to 4 light sources i.e. so the beacons and other light sources that emits light in the infrared spectrum (normal lamps does). Due to this limitation it is not feasible to use the well known 3 or 4 point algorithm or RANSAC for pose estimation. Experiments have shown that approx 2.1 beacons are visible using the Wii interface; In 31% no (true) beacon was visible in the field of view of the camera.

A. Monte Carlo Localisation

As all Bayesian filters, MCL methods address the problem of estimating the state x [13] of a *dynamic system* from measurements or sensor readings. The control theory describes a *dynamic system* as an interaction model between a *controller* and its environment. Both entities interact with each other through *signals* y and *actions* u. The *signal* is taken as an input of the controller and contains observations or measurements of the environment, i.e. observational data such as features extracted from images. The *action* is the output of the controller and is also considered as a measurement, e.g. accelerometers data containing information about Wii motion. The Bayes filter assume that the environment is *Markov*, i.e. past and future data are (conditionally) independent if one knows the current state.
The implementation of a MCL requires two things: the motion model and the sensor model. The motion model is used to integrate the *actions* u to the current pose/state while

the sensor model integrates the observations. The usual MCL algorithm works recursive in four different stages: (1) first, in the *prediction* stage the motion model is used to integrate the *actions* u to all particles e.g. the particles are simply moved. In the following stage (2) the observations are used to *update* the weight π of the particles. Next (3) the weight of all particles is normalized to one. At last (4) the particles are *resampled* to get the posterior distribution. Technically the resampling discards particles with low weights and moves it to a specific (random) particle with a high weight. In our implementation we move to position of the new "offspring" particle in respect to the weight of the parent particle i.e. a low weight of the parent particle will result in a relative high translation. The next section gives the used models that are needed to implement MCL.

B. Motion Model

The motion model $p(x_t|x_{t-1},u_{t-1})$ represents the effects of action u_{t-1} on the Wii's pose (here x_t) in respect to the pose in the last time step. We expect that the Wii is moved by rotation and translation. Our kinematic model assumes the rotational center in the elbow joint of the user, see Figure 3. This assumption enables us to obtain to rotational speeds of roll, pitch and yawn from the accolometers. In experiments we figured out that this assumption ussally holds true: In 90 % of all cases the remote was used properly. To obtain the rotational and translative speeds some addinal work is needed.

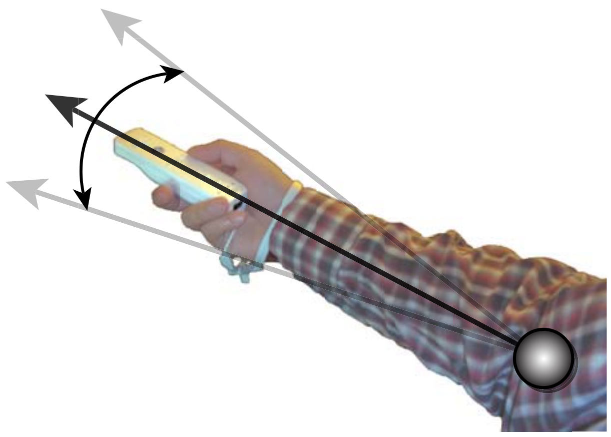


Fig. 3. Kineamtic model of the Wii remote. The rotational center of roll, pitch and yaw is in the elbow joint.

1) Accelerometer data processing:

An accelerometer is measuring acceleration *and* gravity induced reaction forces. One can see that the acceleration can be easily obtained if the influence of the gravity is known. In the case the Nintendo Wii the influence of gravity on the remote is not known due to missing sensors which measures rotational speeds e.g. gyroscopes. We use the particle filter system itself to compensate this: The *believed* pose (including orientation) of each hypothesis is used to guess to *guess* the gravity individually for each particle. Due to the theory of particle filters particles [14] with a proper pose (and orientation) will survive in the sample set and populate the resample set. In theory the set will convert to "good" values after several (> 3) time iterations if beacon are detected. Finally the acceleration can be obtained simple vector algebra.

As next we have to extract the rotational and translative part from the processed acceleration of the Wii. The problem here is that a direct mapping is not possible from a three dimensional vector. For instance the absolute angles of pitch and roll can be easily obtained if the remote is not moved i.e. using the accelerometers output as gravity vector. In the case of a (fast) moving Wii we can only obtain the rotational and translative parts with a certain probability. We use the length of the 3-axis vector $\vec{axel}$ of the raw accelerometers data to calculate the probability τ with a weight function $\tau = gaussianFunction(a|1g-|\vec{axel}||,\mu,\sigma)$. The parameters a,μ,σ are learned with ground truth data. Due to spread of standard factory models the learning process is necessary for all new (not yet unused) Wii remotes. Finally we translate the angles of pitch and roll into rotational speed values.

2) IMM Kalman:

We use the standard Interacting multiple model (IMM) Kalman [15] filter as framework for our motion model. The IMM framework provides for multiple models by allowing distinct process noise models for each of the underlying motion models. The extension to extended Kalman filters with IMM is straightforward. We use two independent models in our implementation: a model for a *moving* remote and a model for a *non-moving* Wii, both first order for translation and rotation. Each particle uses own individual IMM Kalman filters instead of a global IMM filter for all particles.

C. Sensor Model

The *sensor model* is the heart of the MCL. It reflects the probability of measuring y_t if we assume that the state of the Wii is x_t. In our case y_t is the data obtained from the camera sensor. Let m be the number of observations coming from the camera $\Psi = \{\Psi_1,...,\Psi_m\}$ and k be the number of the a-priori known beacons $\Omega = \{\Omega_1,...,\Omega_k\}$. Each Ψ reflects a possible sensor reading of Ω so first a data association is needed. Unfortenatly the problem of a optimal association is NP-hard with known solutions like combinatorial optimization approach e.g. MHT (multiple hypothesis tracking) [16], [17] or sequential Bayesian approach e.g. JPDAF (Joint probabilistic data association filter) [18]. Due to the fact that we use a multi hypothesis tracking system above, it is not needed to track all data associations over time. We use an exhaustive search algorithm ($O(n^2)$) in each time step instead. The algorithm works as follows: First a $\Psi \times \Omega$ matrix of the euclidian error of all possible combinations of $\Psi_{1...m}$ and $\Omega_{1...k}$ is calculated. The euclidian error is the minimal distance of the projected view ray of $\Psi_i|x_t$ to the point $\Omega_l|x_t$. Note that $\Omega_l|x_t$ is ∞ if the beacon is not visible from x_t due to occlusion of the robot itself or angle of beam of the beacon itself. Now the algorithm searches the smallest error in the matrix and assigns the sensor data i exclusively to the beacon j: Ψ_i^j.

We use the following model to calculate the likelihood:

$$p(y_t|x_t) = \sum_{i=0}^{m} p(\Psi_i^j|x_t)p(\Omega_j|x_t) \tag{1}$$

Here it is also assumed that all observations are independent from each other. Here $p(\Psi_i|x_t)$ calculates the probability of every observation with respect to the believed pose (of the particles). In contrast to the approach in literature [19] we are summing up the probabilities instead of multiplying them. This enables a lower sensitivity of (few) "bad" observations to the probability. $p(\Omega_j|x_t)$ reflects the probability that the beacon Ω_j is visible from an specific angle i.e. x_t. We use a gaussian weight function using the angle depending on the type of beacon (quasi point or directional). The likelihood function $p(\Psi_i|x_t)$ is defined as

$$p(\Psi_i|x_t) = \frac{c^2}{c^2 + B(\Psi_i^j, x_t)^2} \tag{2}$$

The function is very similar to the squared error function for errors $B(\Psi_i^j, x_t) \leq c$ and is bounded above by a constant for larger errors, thus the influence of outliers onto the estimate is bounded. We do not use a squared error function due to it is not robust with respect to outliers. In our implementation we choose $c \approx 50$. B is a function which returns the euclidian error from the matrix of Ψ_i^j.

D. Initialisation and Pose estimation

For initialisation the user has to point the Wii remote steady to the centre of the robot. This ensures that sensor data is available and reduces the state space during initialisation. Yaw is zero and roll and pitch can be read out of the accelerometers. The initialisation of the system itself is done in the monte carlo fashion: All hypothesis are randomly distributed in the state space on the x/y/z axis.
In analogy to traditional monte carlo approaches our pose is obtained by building the weighted average of all particles. In the case of no available sensor information of the camera the motion model is only applied to the particles. Here the particles are just moved and not resampled. Due to drift of the sensors we use an additional damping factor of 95% for the motion model if no camera data is available.

VI. Experimental Setup

For our experiments we use a quasi-holonomic mobile robot (Fig. 4) "James". James is based on the BlueBotics platform "Movement" [20] and is equipped with an SICK LMS 200 laser range finder mounted to its front and two stereo cameras for navigation. With a payload of 150 kg, the platform has a maximum speed of 5km/h.

We mounted four quasi point-like ir- beacons and three directional ir- beacons on James. All beacons are visible if the remote is places in front of the robot and points to its centre. The quasi point-like on top of the setup are 170 degrees visible from the front of the robot. This is due to construction issues of the robot itself. The other quasi points like beacons on bottom are almost visible from all sides of the robot. The MCL prototype is implemented in Matlab on a Laptop with 1.6Ghz Pentium M. and runs with 5Hz with 40 particles in the normal use and 200 particles in the initialisation phase.

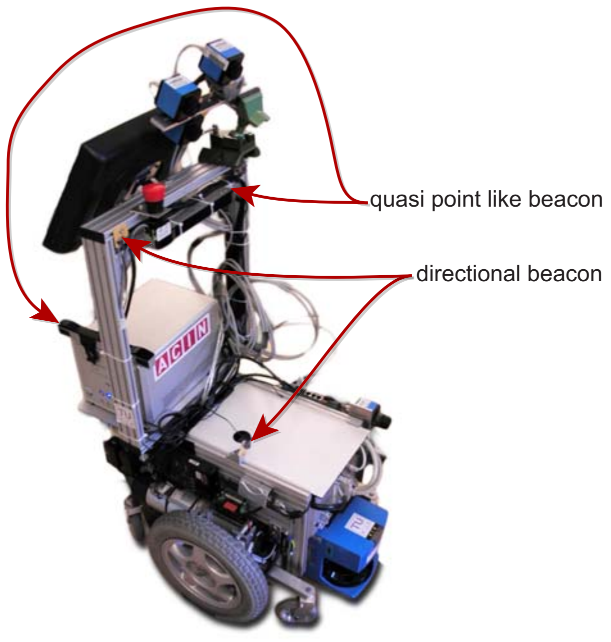


Fig. 4. Implemention of the interface on our robot James

VII. User Trails and Results

In this section we describe the evaluation of the Wii "virtual leach" interface with a reprehensive group of volunteers. The main concept of the evaluation is to let a volunteer to guide the robot to three specific spots in a test living room (Fig. 5) in a specific order (1 to 2, 2 to 3 and 3 to 1) as fast as possible using the Wii interface. After a short instruction the volunteer had to guide the robot without any help of the instructor. For the sake of simplification we place the robot at spot no 1 as start spot for all trails. The environment itself and spots are identical for all volunteers.

The test environment consists at of two rooms with a driveable area of $15m^2$. The room on the left-hand side is a typical living room with at two seating-accommodation; The right room is a workroom. The room layout offers enough space for three non overlapping "spots of interest":

1) A start spot that is easy to access.
   - min 1.5m to the margins
2) A second spot that is difficult to reach and to leave.
   - 0.2m - 0.3m to the margins left and right, min 1.5m to the front
3) A third spot that is close to solid furniture and provides a (partial) different type of ground surface e.g. carpet.
   - max 0.3m to the furniture in front, 0.5m-1m left and right

The different margins of each spot are defined on purpose. The first spot is easy to give the volunteer a chance to

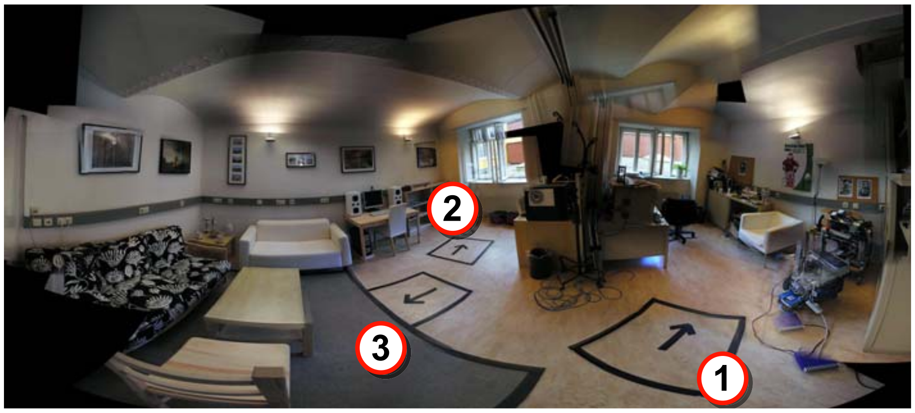
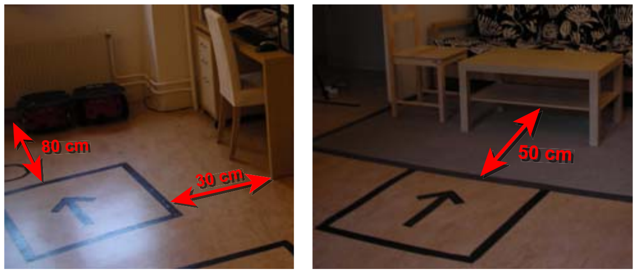


Fig. 5. Panoramic view of the used test environment. Three spots are marked with a black 1x1m square. The arrow shows the required orientation of the robot for each spot.



(a) Spot 2 (b) Spot 3

Fig. 6. Details of the environment

get a ”feeling” for the interface and robot. The second spot requires advanced navigation skills to move the robot through corridor (Fig. 6(a)). The narrowness of the corridor has also another purpose: It is too tight to fit a human and robot at the same time. This prevents the volunteer to pass the robot while it’s moving though the corridor. As a side effect the volunteer has only two chances to leave the place with the robot: The robot must be turned 180 degree or ”pushed back”. The idea behind spot three is similar to the previous one: The place does not offer enough space in front of the robot (Fig. 6(a)) for the volunteer to reach the spot. At least 1.2m free space to the furniture would be needed to reach the spot. Please note that the user must be in front of the robot. It is on purpose that the user has to ”move around / move behind” the furniture first. In our setup we used a small coffee table on a carpet. The alignment of the carpet relative to the spot is again on purpose: The robot has to turn to the left to reach the start position; it is likely that at least one wheel1 runs on the carpet while the other wheels still run on the regular ground. In our setup the carpet provides a much better grip than the (slippery) normal ground. Now the robot is more difficult to control.

We asked a group of 12 volunteers to use the Wii interface for evaluation. The mean age of the group is 30 years old (i.e. 21-38). The overall experience using robots is ranging form ”first time I touches a robot” to ”I know how use a remote controlled toy car”. Almost all volunteers consider themselves ”open to new technologies”. First, every volunteer was instructed who to use the interface and the robot (approx 30sec), by the typical learning-by-doing approach. Then the user had to control the robot to all spots in the specific order 1 to 2,2 to 3 and 3 to 1 with no help from the instructor. A spot is considered as reached if:

- The robot is surrounded by the 1m x 1m black square on the ground (see Fig. 5)
- No part of the robot is touching the black square
- The orientation of the robot is the desired orientation of the spot
- The robot is inside the area for at least 2 seconds

Every user was allowed to do a restart or try a better/faster trail with the robot once. All volunteers were able to guide the robot through the environment with ”almost” no touching of the environment during the first run. All volunteers were willing to try a second trail; a restart was not needed by any user. The second run was almost 20% faster than the first one.

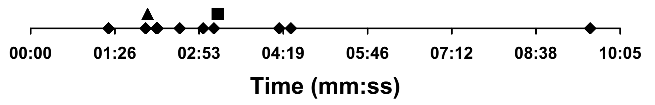


Fig. 7. Results of the User Trails. The fastest run per volunteer is counted. The diamonds represents the Wii interface, the square the average time for a touch-panel interface and the triangle average the time of a joystick interface

Figure 7 shows the time for the fastest run per volunteer. The time for a touch panel and joystick interface are provided for the sake of completeness. One can see that the average time is in good company with the touch panel and joystick interface.
As expected the corridor of spot no. 2 was the most difficult part of the environment. No user tried to ”push back” the robot to leave the spot and move to spot no 3. All tried to turn the robot ”on-spot”. The two different types of ground was no problem for the most users. In fact all users were

surprised that the robot reacts total different with two types of ground with different grip (see above).

Two volunteers underestimated the speed of the robot and hit slightly the corridor of spot no. 2. Another volunteer managed to bend a metal trashcan by moving the robot backwards. One user stated: "I did not understand the concept behind the interface, but its fun". 10 of 12 liked to interface, the rest was unsure. All in all the interface was well accepted by the users.

VIII. Conclusions

In this paper we presented an approach for an inexpensive interface for Robots using the Nintendo Wii Remote. It is build from inexpensive hardware i.e. the Nintendo Wii Controller and a couple of infrared LEDs. The needed computational power for computation is manageable and adjustable trough the number of hypothesis. We were surprised that the interface was quickly accepted by most of the users (Fig. 8).

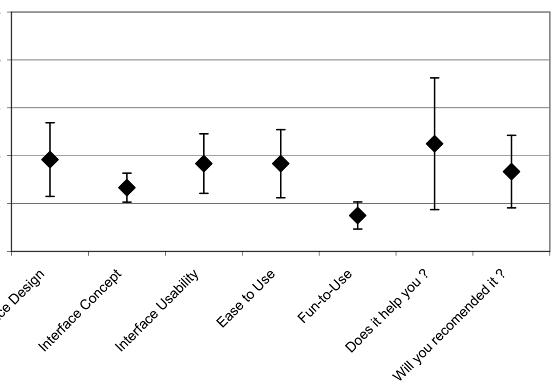


Fig. 8. Results of the User Interviews. "1" is best, "6" is worse.

First implementations of the leach (the robot & Wii was connected through a leach) have shown that a real leach is inefficient: The leach was almost wrapping the robot after a short amount of time. It turned out that a virtual-leach is much easer to handle, but it is difficult to understand. One user complained the missing haptics-feedback of the virtual leach compared to a real leach which is a clear drawback of our approach. Another drawback is the limited accuracy of the accelerometers inside the Wii. Slow motions ($< 1.5\frac{cm}{s}$) are not recognized by these sensors. We plan to extend our approach by the usage of cyclic switch on/off ir-beacons running at different a-priori known frequencies e.g. 1Hz and 0.33Hz to improve the robustness to false positives. The accuracy of our approach can be improved using the Wii-motion plus.

IX. Acknowledgments

The research leading to these results has received funding from the European Community's Sixth Framework Programme (FP6/2003-2006) under grant agreement FP6-2006-IST-6-045350 (robotshome). We thank all our participants who spent their time and efforts for the user trials. The authors gracefully acknowledge the support of Peter Einramhof and Manuel Pascual.

References

[1] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.

[2] Michael Isard and Andrew Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[3] Walterio Mayol, Andrew J, Davison, Ben Tordoff, Nick Molton, and David W. Murray. Interaction between hand and wearable camera in 2d and 3d environments. In *Proceedings of British Machine Vision Conference (BMVC)*, 2004.

[4] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, F. Fox, D. Hähnel, G. Lakemeyer, C. Rosenberg, N. Roy, J. Schulte, D. Schulz, and W. Steiner. Experiences with two deployed interactive tour-guide robots. In *Proceedings of the International Conference on Field and Service Robotics*, Pittsburgh, PA, 1999.

[5] Charles C. Kemp, Cressel D. Anderson, Hai Nguyen, Alexander J. Trevor, and Zhe Xu. A point-and-click interface for the real world: laser designation of objects for mobile manipulation. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction (HRI)*, 2008.

[6] J. A. Corrales, F. A. Candelas, and F. Torres. Hybrid tracking of human operators using imu/uwb data fusion by a kalman filter. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction (HRI)*, 2008.

[7] Matthias Rehm, Nikolaus Bee, and Elisabeth André. Wave like an egyptian Ů accelerometer based gesture recognition for culture specific interactions. In *Procedings of HCI 2008 Culture, Creativity, Interaction*, 2008.

[8] Christine Connolly. Kuka robotics open architecture allows wireless control. *Industrial Robot: An International Journal*, 35(1):12–15, 2008.

[9] Cheng Guo and Ehud Sharlin. Exploring the use of tangible user interfaces for human-robot interaction: a comparative study. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, 2008.

[10] T. Pintaric and H. Kaufmann. Affordable infrared-optical pose tracking for virtual and augmented reality. In *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, 2007.

[11] Timo Pylvänäinen. *Pattern Recognition and Image Analysis*, chapter Accelerometer Based Gesture Recognition Using Continuous HMMs. Springer Berlin / Heidelberg, 1th edition, 2005.

[12] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localisation: Efficient position estimation for mobile robots. In *the National Conference on Artificial Intelligence*, 1999.

[13] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[14] Dan Crisan. Particle filters - a theoretical persperctive. In Arnaud Doucet, Nando de Freitag, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Parctise*, pages 17–41. Springer New York, 2001.

[15] H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33(8):780–783, Aug 1988.

[16] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transaction on Automatic Control 24(6)*, pages 843–854, 1979.

[17] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and data association*. Mathematics in science and engineering ; 179. Boston, first edition, 1988.

[18] D. Schulz, W. Burgard, D., Fox, and A.B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. In *International Journal of Robotics Research (IJRR)*. Springer, 2003.

[19] Sebastian Thrun, Dieter Fox, and Wolfram Burgard. Monte carlo localization with mixture proposal distribution. In *AAAI/IAAI*, pages 859–865, 2000.

[20] P. Mayer, G. Edelmayer, G.J. Gelderblom, M. Vincze, P. Einramhof, M. Nuttin, T. Fuxreiter, and G. Kronreif. Movement -modular versatile mobility enhancement system. In *Procedings of IEEE International Conference on Robotics and Automation (ICRA), Rome*, 2007.