The 2016 AAAI Fall Symposium Series:
Shared Autonomy in Research and Practice
Technical Report FS-16-05

Detecting User Intention Changes Using the Kullback-Leibler Distance

Eric Demeester and **Alexander Hüntemann**
Department of Mechanical Engineering, Faculty of Engineering Technology, Research group ACRO
KU Leuven - University of Leuven, Technology Campus Diepenbeek
Diepenbeek, Belgium
eric.demeester@kuleuven.be

Abstract

Many people may benefit from assistive robots that understand their users' intentions and aid them with the execution of these intentions in a safe and intuitive way through shared control. In the past, our research group has worked on semi-autonomous robotic wheelchairs transporting people with mobility challenges. Experimental results with our user-adaptive Bayesian approach for both intention estimation and shared human-machine decision-making under uncertainty have shown that in situations where the driver changes his or her intention, the assistive behavior by the robot may under certain conditions be counter-intuitive as it continues to take actions that are in line with the previous user intention, and this for too long a period of time. To remedy this, this paper proposes an approach to detect such changes in user plans in order to make the robot's assistive behavior more reactive and thus more intuitive. The approach adopts a test that checks the consistency of the posterior distribution over user intentions with the given steering signals. A proof-of-concept study of this test's performance is shown.

Introduction

One of the major distinctions between humans and most animals may lie in the human's construction and usage of tools. As an example, robots were developed as tools to assist humans in dangerous, heavy-duty or repetitive tasks. One of the many types of assistive robots are robotic wheelchairs, which are developed to reduce the mobility challenges of a large group of elderly and physically impaired people, thereby also decreasing the number of accidents and increasing these people's independence from others and quality of life in general. Many research groups have spent substantial resources on this noble goal, see e.g. (Simpson and Levine 1999), (Yanco 2000), (Montesano et al. 2010), (Parikh et al. 2004), (Lankenau and Röfer 2001), (Carlson and Millán 2014).

An *effective* assistive robot should understand the plans of its user and aid him or her with the execution of these intentions in a safe and intuitive way through shared control. In order to realize this, our research group has devised and tested a user-adapted Bayesian approach for both intention estimation and joint human-machine decision-making under uncertainty. However, experimental results with this framework have shown that in situations where the driver changes his or her intention, the assistive behavior by the robot may under certain conditions be counter-intuitive as it continues to take actions corresponding to the previously estimated intention for too long a period of time. This paper proposes an approach to detect such changes in user intention in order to make the robot's assistive behavior more reactive and thus more intuitive. The approach adopts a test that checks the consistency of the posterior distribution over user intentions with the given steering signals.

This paper is organized as follows. First, our approach to Bayesian intention estimation is briefly reviewed. Then, an approach to detect user intention changes based on the Kullback-Leibler distance is proposed. Next, an experimental evaluation of this consistency test is discussed, before concluding this paper and pinpointing future work.

Framework for Bayesian Intention Estimation

Our Bayesian plan recognition approach is briefly discussed in this section. Demeester et al. and Hüntemann et al. provide further details.

Notation

Consider the control-theoretic view on shared control and plan recognition shown in Figure 1. A human operator

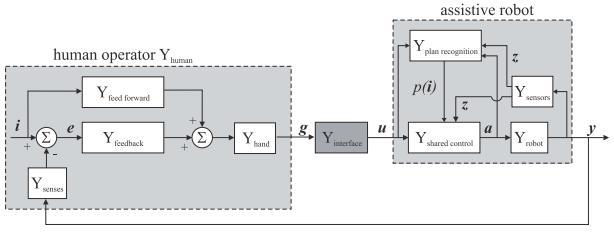

Figure 1: This control-theoretic view on plan recognition and shared control introduces the adopted notation.

Y_{human} gives user signals $\boldsymbol{g}$ to control the assistive robot Y_{robot}, where $\boldsymbol{g}$ represent the measured signals such as potentiometer voltages in traditional joysticks or EEG. Interface signals $\boldsymbol{u}$ can be either discrete or continuous, and can

be sampled or occurring at asynchronous time instants. In case the user signals $\boldsymbol{g}$ are directly observed, as is the case for most current interfaces, $\boldsymbol{u}$ equals $\boldsymbol{g}$ or $Y_{interface} \equiv 1$. The operator has been simplified in this scheme as an element that reacts upon an error signal $\boldsymbol{e}$, which stems from the difference between the system's desired behavior or reference signal $\boldsymbol{i}$ and the system's actual behavior $\boldsymbol{y}$ as observed through the human's senses Y_{senses}. $Y_{feed\ forward}$ represents internal models the user has of the system.[1] The human's desired control signal can possibly be corrupted by physical handicaps Y_{hand}. Plan or intention recognition in a sense boils down to estimating the reference trajectory $\boldsymbol{i}$ the user has in mind for the robot. The shared controller $Y_{shared\ control}$ receives signals $\boldsymbol{z}$ from all kinds of sensors $Y_{sensors}$, user signals $\boldsymbol{u}$, and a probability function over possible states $\boldsymbol{i}$ from the plan recognition module $Y_{plan\ recognition}$ in order to calculate a decision or action $\boldsymbol{a}$ while dealing with uncertainty. Except when ground truth is known such as in simulations, the shared controller observes the robot's output or behavior $\boldsymbol{y}$ only through its sensors $Y_{sensors}$. This means that $\boldsymbol{y}$ is not always available, but only indirectly observable as $\boldsymbol{z}$. The sensor measurements $\boldsymbol{z}$ will not always be directly adopted, but may be transformed into an estimation of the robot's surroundings, its pose $\boldsymbol{p}_{robot}$ in this map, and its twist $\boldsymbol{t}_{robot}$. The plan recognition module $Y_{plan\ recognition}$ estimates the state or intention $\boldsymbol{i}$ based on the user's output $\boldsymbol{u}$ and the user's input $\boldsymbol{y}$. All user signals up to discrete time index k are referred to as $\boldsymbol{u}_{1:k} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k\}$, all sensor measurements as $\boldsymbol{z}_{0:k-1} = \{\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{k-1}\}$, all shared control actions as $\boldsymbol{a}_{0:k-1} = \{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{k-1}\}$, and all robot behavior as $\boldsymbol{y}_{0:k-1} = \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{k-1}\}$. The user plan evolution till time k is referred to as $\boldsymbol{i}_{1:k} = \{\boldsymbol{i}_1, \ldots, \boldsymbol{i}_k\}$.

Bayesian Estimation of User Plans

At time k, $\boldsymbol{u}_{1:k}$, $\boldsymbol{z}_{0:k-1}$ and $\boldsymbol{a}_{0:k-1}$ are known to the plan recognition module. Furthermore, a priori information $\boldsymbol{b}_{0:k}$ such as known parameters in models or maps of the environment may be available. For notational simplicity, $\boldsymbol{b}_{0:k}$ is omitted in the formulas below (all probabilities are conditioned on $\boldsymbol{b}_{0:k}$). Bayesian plan recognition then proceeds as follows:

$$
\begin{aligned}
& p_k\left(\boldsymbol{i}_{1:k} | \boldsymbol{u}_{1:k}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\
& \overset{\text{Bayes' rule}}{=} p_{user}\left(\boldsymbol{u}_k | \boldsymbol{i}_{1:k}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\
& \quad \cdot p_{k-1}\left(\boldsymbol{i}_{1:k} | \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \cdot \eta \\
& \overset{\text{product rule}}{=} p_{user}\left(\boldsymbol{u}_k | \boldsymbol{i}_{1:k}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\
& \quad \cdot p_{process}\left(\boldsymbol{i}_k | \boldsymbol{i}_{1:k-1}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\
& \quad \cdot p_{k-1}\left(\boldsymbol{i}_{1:k-1} | \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \cdot \eta \\
& \overset{\text{causal system}}{=} p_{user}\left(\boldsymbol{u}_k | \boldsymbol{i}_{1:k}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\
& \quad \cdot p_{process}\left(\boldsymbol{i}_k | \boldsymbol{i}_{1:k-1}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\
& \quad \cdot p_{k-1}\left(\boldsymbol{i}_{1:k-1} | \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-2}, \boldsymbol{a}_{0:k-2}\right) \cdot \eta
\end{aligned}
\tag{1}
$$

[1] Reality is probably much more complex. It is not the purpose of this scheme to present a truthful model of a human operator though.

where η is a scalar normalisation factor to ensure that the probabilities over all user plans sum up to 1.

Our shared control algorithms currently only require knowledge of $p\left(\boldsymbol{i}_k | \boldsymbol{u}_{1:k}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right)$, which is obtained by marginalizing over $\boldsymbol{i}_{1:k-1}$:

$$
\begin{aligned}
& p\left(\boldsymbol{i}_k | \boldsymbol{u}_{1:k}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\
& = \sum_{\boldsymbol{i}_{1:k-1}} p\left(\boldsymbol{i}_{1:k} | \boldsymbol{u}_{1:k}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right)
\end{aligned}
\tag{2}
$$

We have given the likelihood function $p_{user}\left(\boldsymbol{u}_k | \boldsymbol{i}_{1:k}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right)$ the name *user model*, because it yields the likelihood that the user outputs signals $\boldsymbol{u}_k$ given that he or she has had intention evolution $\boldsymbol{i}_{1:k}$, given all past user signals $\boldsymbol{u}_{1:k-1}$, and given all past system behavior encoded as $\boldsymbol{a}_{0:k-1}$ and environmental changes encoded as $\boldsymbol{z}_{0:k-1}$. In a way, the user model predicts the user signals. Function $p_{process}\left(\boldsymbol{i}_k | \boldsymbol{i}_{1:k-1}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right)$ is referred to as the *plan transition* or *plan process model*. It models the dynamics of user plans, i.e. how user plans $\boldsymbol{i}_k$ evolve over time given previous plans $\boldsymbol{i}_{1:k-1}$, past user signals $\boldsymbol{u}_{1:k-1}$, environmental changes $\boldsymbol{z}_{0:k-1}$ and system behavior $\boldsymbol{a}_{0:k-1}$. Factor p_{k-1} in Equation 1 equals the prior probability function, i.e. the probability function at time $k-1$. Equation 2 as such is not recursive: it requires to retain and adopt all previously gathered information to make new estimates regarding user plans. However, updates can be made recursive by assuming that information from at most m past time steps influences the current user plan and user signal, see (Demeester et al. 2008).

Application to Robotic Wheelchairs

The described Bayesian approach to intention estimation still leaves many implementation choices open, such that it can be tailored to a specific application. For robotic wheelchairs, it can be assumed that the driver desires to drive to some end pose $\boldsymbol{p}_e = [x_e\ y_e\ \theta_e]^T$ with a certain end twist $\boldsymbol{t}_e = [v_e\ \omega_e]^T$, where θ_e represents the robot orientation at the goal position $[x_e\ y_e]^T$, and where v_e denotes the desired linear velocity and ω_e the desired rotational velocity at $\boldsymbol{p}_e$. A velocity or twist $\boldsymbol{t}$ and pose $\boldsymbol{p}$ will be represented jointly as the robot state $\boldsymbol{x}$. A user plan or intention $\boldsymbol{i}_k$ at time k can then be generically described as a trajectory that the user has in mind to achieve the goal state $\boldsymbol{x}_{goal}$ from the current robot state $\boldsymbol{x}_{current}$. This trajectory can amongst others be represented as a sequence of robot states:

$$
\boldsymbol{i}_k = \{\boldsymbol{x}_{current}, \cdots, \boldsymbol{x}_{goal}\}. \tag{3}
$$

Hypotheses regarding user plans $\boldsymbol{i}_k$ can be generated in a variety of ways, e.g. by first generating all plausible goal state candidates $\boldsymbol{x}_{goal}$ based on knowledge of the robot's surrounding environment, and then all trajectories to these goal states. These trajectories $\{\boldsymbol{x}_{current}, \cdots, \boldsymbol{x}_{goal}\}$ can be calculated using a motion planning algorithm. In order to be able to recognize also complex user plans such as docking and parking maneuvers, the motion planner should take the robot's kinematics, orientation and geometry accurately into account.

Dealing with Multiple Intent Hypotheses

The probability distribution p_k can be represented in various ways, and may not always be consistent with the actual state. For this, lessons can be learned from robot localization approaches. Grid-based Markov Localization by (Fox, Burgard, and Thrun 1998) discretizes the state space of robot poses. In order to focus computational resources more towards regions where it is needed, others have used a mixture of Gaussians to model multiple robot pose hypotheses such as (Jensfelt and Kristensen 2001), or sets of particles such as (Lenser and Veloso 2000). After an initial phase where there is global pose uncertainty, these approaches gradually focus on a limited set of hypotheses until there is convergence to a single mode in the probability function. However, the robot pose may be lost again afterwards, for example by kidnapping the robot and putting it at a random pose. In that case, after *detection* that localization is lost, *dealing with unknown robot poses* is either performed by localizing again from scratch by imposing a uniform distribution over the state space such as by Porta, Verbeek, and Kröse, by adding additional samples or pose hypotheses to the existing set such as by Fox, or by replacing existing samples with more likely samples, see e.g. (Lenser and Veloso 2000).

For assistive robots, the kidnapped robot problem corresponds to a sudden change in user plan when the plan recognition algorithm is strongly focused on another user plan. At all times, it should be verified that the actual user intention has not changed, otherwise incorrect assistance will be provided. For this, consistency tests such as the one discussed next will prove useful.

Detection of User Intent Changes

In order to effectively deal with intent changes, we have combined two approaches: (1) ensuring that all user intentions have at all times a minimum probability, and (2) testing the consistency of the maintained set of user intention hypotheses with the given user signals. This section discusses the adopted algorithms behind both approaches.

Ensuring a Minimum Probability

User plan hypotheses that are assigned zero probability at one point in time will never obtain a non-zero probability afterwards in the Bayesian framework. Furthermore, for plan hypotheses with a very small probability, it takes some time to reach a certain probability level again. Therefore, some researchers such as Larsen advise to avoid probabilities near 0 or 1 in Bayesian statistics, to prevent singular behavior for plan hypotheses with probabilities around zero. Ensuring a minimum probability can be compared to the injection of noise in system models of Kalman filters, or to the replacement of some samples by random samples in particle filters such as by Lenser and Veloso, prior to performing the observation update step.

For these reasons, we want to ensure a minimum probability $\alpha_{\texttt{min}}$ for all plan hypotheses. Various approaches for ensuring this minimum probability can be devised. The basic idea behind approaches implemented in this work is depicted in Figure 2 for discrete probability functions. The

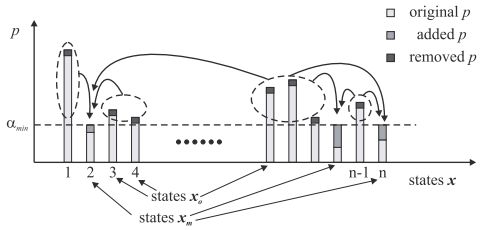

Figure 2: The basic idea behind approaches to ensure a minimum probability for each state $\boldsymbol{x}$. The m states $\boldsymbol{x_m}$ that have a probability below a threshold $\alpha_{\texttt{min}}$, are assigned $\alpha_{\texttt{min}}$. The required probability mass for this originates from some or all states $\boldsymbol{x_o}$ that have *enough* probability. These states $\boldsymbol{x_o}$ may or might not contribute equally to the requested probability mass.

pseudo code of two possible variants is shown in Algorithms 1 and 2. Algorithm 1 assumes that the probability

Algorithm 1 Ensuring a minimum probability for a discrete probability function over n states, using subtraction.

```
1: for i ← 1 to n do
2:     hasMinBelief(i) ← false
3: m ← 0 {number of states with minimum probability}
4: repeat
5:     sum_p ← 0
6:     for i ← 1 to n do
7:         if p_i < α_min then
8:             sum_p ← sum_p + α_min − p_i
9:             p_i ← α_min
10:            hasMinBelief(i) ← true
11:            m ← m + 1
12:    if sum_p ≠ 0 then
13:        for i ← 1 to n do
14:            if hasMinBelief(i) ≡ false then
15:                p_i ← p_i − sum_p/(n−m)
16: until sum_p ≡ 0
```

function has been normalised prior to running the algorithm. It first checks which states $\boldsymbol{x_m}$ have a probability below the minimum $\alpha_{\texttt{min}}$ (line 7). These states are assigned the minimum probability (line 9). Array $\boldsymbol{hasMinBelief}$ keeps track of the m states that have been assigned $\alpha_{\texttt{min}}$. The probability mass sum_p that is required to guarantee a minimum probability for these m states is obtained from all other states $\boldsymbol{x_o}$ having a probability $p > \alpha_{\texttt{min}}$. These states contribute equally to sum_p, as the same probability mass is subtracted from each $\boldsymbol{x_o}$ (line 15). This subtraction may result in new states having too low a probability. Therefore, the algorithm iterates until no new states with too low a probability are found. Algorithm 2 on the other hand does not assume the probability function to be normalized before running the algorithm. It checks for each state $\boldsymbol{x_i}$ whether its probability p_i would be below threshold $\alpha_{\texttt{min}}$ after normalization of its current value v_i (lines 10-11). Normalization of a state's value v_i is done by multiplication of v_i with a scale

Algorithm 2 Ensuring a minimum probability for a discrete probability function over n states, using multiplication.

1: $sum_v \leftarrow 0$
2: **for** $i \leftarrow 1$ to n **do**
3: $\quad \boldsymbol{hasMinBelief}(i) \leftarrow$ false
4: $\quad sum_v \leftarrow sum_v + v_i$
5: $m \leftarrow 0$ {number of states with minimum probability}
6: **repeat**
7: $\quad m_{new} \leftarrow 0$ {number of new states with too low a probability mass}
8: $\quad$ **for** $i \leftarrow 1$ to n **do**
9: $\quad\quad$ **if** $\boldsymbol{hasMinBelief}(i) \equiv$ false **then**
10: $\quad\quad\quad p_i \leftarrow \frac{(1-m\cdot\alpha_{\texttt{min}})\cdot v_i}{sum_v}$
11: $\quad\quad\quad$ **if** $p_i \leq \alpha_{\texttt{min}}$ **then**
12: $\quad\quad\quad\quad sum_v \leftarrow sum_v - v_i$
13: $\quad\quad\quad\quad p_i \leftarrow \alpha_{\texttt{min}}$
14: $\quad\quad\quad\quad \boldsymbol{hasMinBelief}(i) \leftarrow$ true
15: $\quad\quad\quad\quad m \leftarrow m + 1$
16: $\quad\quad\quad\quad m_{new} \leftarrow m_{new} + 1$
17: **until** $m_{new} \equiv 0$
18: **for** $i \leftarrow 1$ to n **do**
19: $\quad$ **if** $\boldsymbol{hasMinBelief}(i) \equiv$ false **then**
20: $\quad\quad p_i \leftarrow \frac{(1-m\cdot\alpha_{\texttt{min}})\cdot v_i}{sum_v}$

factor a, which is obtained as follows. Suppose there are m states $\boldsymbol{x_m}$ with minimum probability $\alpha_{\texttt{min}}$, and $o = n - m$ other states $\boldsymbol{x_o}$ with value v_o that have not been normalized yet. The normalized values p_o of states $\boldsymbol{x_o}$ should adhere to: $m \cdot \alpha_{\texttt{min}} + \sum_o p_o = 1$. Since $p_o = a \cdot v_o$, a equals:

$$a = \frac{1 - m \cdot \alpha_{\texttt{min}}}{\sum_o v_o}. \tag{4}$$

If $p_i \leq \alpha_{\texttt{min}}$, the state is assigned the minimum probability (line 13). If no states are found anymore with too low a probability mass, the remaining $n - m$ states that were not assigned the minimum probability are normalized (lines 18-20).

A simple experiment shows that undesired effects indeed occur for probabilities very close to zero. Figure 3 (a) shows a discrete a priori distribution f and a discrete observation function g over a certain state space x. The a priori distribution is initialized by assigning to the $n = 50$ bins a probability mass p_n equal to

$$p_n = \frac{1}{\sigma_f\sqrt{2\pi}} \exp\left(-\frac{(x_n - \mu_f)^2}{2\sigma_f^2}\right) \tag{5}$$

where μ_f and σ_f are respectively the mean and standard deviation of the Gaussian that is used to initialize the a priori function f, and x_n represents the $x-$position of the center of the nth bin. After this initialization, the discrete distribution is normalized. The domain of the probability function equals $x \in [0, 15]$. For the observation function g, a similar initialization with a Gaussian is adopted, but with a mean μ_g and a standard deviation σ_g. The observation function is adopted for $k = 100$ times to update distribution f using Bayes' rule. For $k \to \infty$, this should result in a single peak at the mean of the observation function. Figure 3 (b) shows the evolution of the probability function f over time if no minimum probability is ensured. The darker a cell, the more probable it is. Figure 3 (c) shows the evolution of the probability function f over time if a very low minimum probability is ensured of $\alpha_{\texttt{min}} = 10^{-10}/n$. As can be seen, ensuring that all probabilities are larger than a threshold $\alpha_{\texttt{min}}$ avoids the singular behavior for probabilities near zero and allows f to converge much faster to a position around μ_g, even if $\alpha_{\texttt{min}}$ is very low. Values of $\alpha_{\texttt{min}}$ as low as $10^{-30}/n$ yield similar results, although convergence is more delayed. However, as the mean of f approaches the mean of g, the effect of ensuring a minimum probability is decreased, which is shown in Figures 4 (a) and (b): convergence of f to g is slower. Increasing the threshold further to $\alpha_{\texttt{min}} = 0.01/n$ has a positive effect on convergence in these cases, and this threshold may be considered to not destroy much of the previously gathered information.

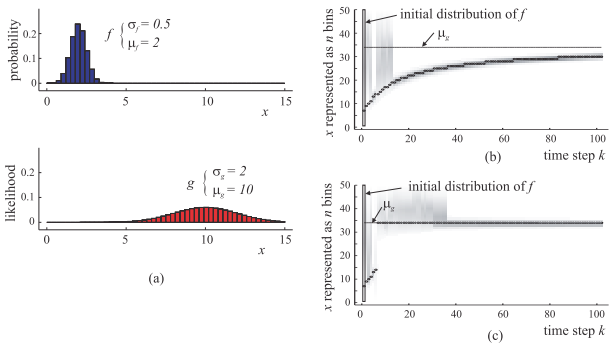

Figure 3: Figure (a) shows the initial discrete distribution of f and the observation function g. Figure (b) shows the evolution of f when updating f for $k = 100$ times using Bayes' rule with g as the observation function. No minimum probability $\alpha_{\texttt{min}}$ is adopted, and f converges very slowly to a position around μ_g. Darker cells indicate states x that are more probable. Figure (c) shows that this convergence is much faster if a minimum probability threshold (in this case $\alpha_{\texttt{min}} = 10^{-10}/n$) is adopted.

Kullback-Leibler Consistency Test

Consistency tests are adopted to check if a running hypothesis is still consistent with new observations. Inconsistent hypotheses typically originate from an incorrect state estimate, an incorrect observation model (p_{user}), or an incorrect system model ($p_{process}$). Various consistency tests have been used for global localization. Jensfelt and Kristensen adopt a NIS test for checking if new sensor observations match with one of the robot pose hypotheses in their multi-hypothesis localisation algorithm. Lenser and Veloso compare the average likelihood of all samples given the observations to a user-defined threshold. Porta, Verbeek, and Kröse check if the sum of the likelihood of the particles exceeds a user-defined threshold for a user-defined number of consecutive times. Also outside the field of robot localization, various

techniques exist to check whether two histograms are instantiations of the same true probability function. Narsky and Porter provide an overview.

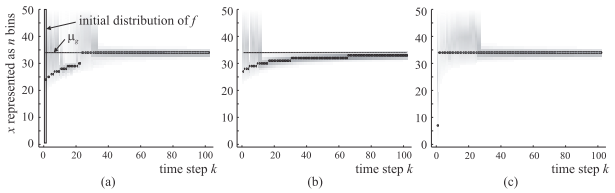

Figure 4: Figure (a) and (b) depict the same situation as in Figures 3 (a) and (c), but with as means for f respectively $\mu_f = 7$ and $\mu_f = 8$. Figure (c) depicts the same situation as in Figures 3 (a) and (c), but with application of a consistency test with threshold $\alpha_{\texttt{KLD}} = 8$. Right from time step 1 it is detected that f and g are not consistent, and f is uniformized and initialized with g by applying Bayes' rule once. An almost instant convergence to the true state is obtained.

Since in this work no single user plan estimate $\boldsymbol{i}$ will be chosen as the true hypothesis, the compatibility of the complete probability function needs to be compared somehow to new measurements (user signals $\boldsymbol{u}$). For this, the symmetric Kullback-Leibler distance (KLD) will be adopted to compare the probability function p_{pred}, which predicts the probability of user intentions in the next time step:

$$\begin{aligned} p_{pred} \quad = \quad & p_{process}\left(\boldsymbol{i}_k|\boldsymbol{i}_{1:k-1}, \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-1}, \boldsymbol{a}_{0:k-1}\right) \\ & \cdot p_{k-1}\left(\boldsymbol{i}_{1:k-1}|\boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{0:k-2}, \boldsymbol{a}_{0:k-2}\right) \end{aligned}$$

with the likelihood function p_{user} of the latest user signal:

$$D\left(p_{pred}||p_{user}\right) + D\left(p_{user}||p_{pred}\right) \tag{6}$$

where:

$$D\left(p_1||p_2\right) = \sum p_1 \cdot \log \frac{p_1}{p_2} \tag{7}$$

If this measure exceeds a threshold $\alpha_{\texttt{KLD}}$, an inconsistency between the user's steering signals and the robot's belief over user intentions is assumed to be present. In that case, the a priori distribution p_{k-1} is re-initialized to a uniform distribution, and user plans are estimated from scratch again. Figure 4 (c) illustrates this approach for the same parameters for f and g as in Figure 3 (a). The Kullback-Leibler distance threshold was set to 8. The incorrectly estimated state is detected in the first time step, the probability function is re-initialised with a uniform distribution, and convergence to the correct state is almost immediate. However, for Figures 4 (a) and (b), no inconsistencies were detected using this threshold. The Kullback-Leibler distance depends on the number of states (intentions), which may vary over time. This may complicate the choice of an accurate, stable threshold α_{KLD}. The next section will discuss this further.

Experimental Results

Figure 5 (left) defines the adopted experiment to evaluate user intent change detection. The user starts in the middle of a circle of possible goal locations. Using a switch interface containing 3 buttons (left, right, forward) the driver tries to execute straight-line paths to different goal locations positioned on a circle, where the driver is either in full control of the wheelchair, or gets shared control assistance using the Greedy POMDP algorithm described by (Demeester 2014). Whenever the driver reaches a goal location within a radius of $r_{goal} = 0.35$ m, another goal location is chosen to which the driver should go. The desired driving direction is chosen and indicated to the user by the simulator, but the intention estimation algorithm has no access to this ground-truth knowledge on the user's intent. The adopted user model p_{user} is modeled to depend only on the relative angle of the goal position w.r.t. the wheelchair, as shown in Figure 5 (right), see also (Demeester 2014). Intent changes occur whenever the driver reaches a goal location. Furthermore, while driving to certain goal locations, sudden intent changes are introduced by the simulator.

Prior to performing the tests, α_{min} and α_{KLD} should be determined. The KLD threshold α_{KLD} is dependent on α_{min}. We would like to find a way to automatically determine an appropriate threshold α_{KLD}. One can devise the following upper and lower bound to this threshold. Figure 6 (a) shows the case where we do not want our algorithm to find an inconsistency: this is the case where our probability function over intentions p_{pred} is uniform and the user model p_{user} indicates a single intent as the most likely intent. Vice versa, if the user model is uniform and the probability function has all probability mass focused on one intent, no inconsistency should be found. Remark that we ensured a minimum probability both for p_{pred} and p_{user}, in this case with $\alpha_{min} = 0.01/n$ (n representing the number of intents). As a result, $\alpha_{KLD} \geq 10.64$ if $n = 24$. Figure 6 (b)

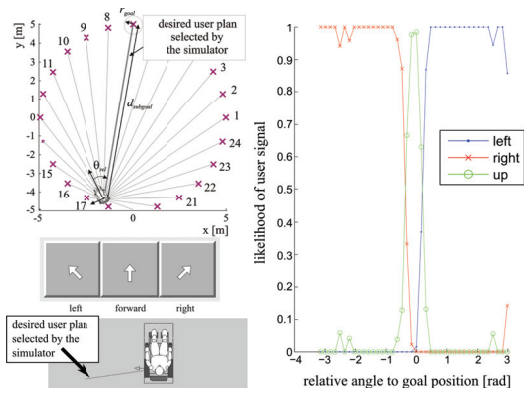

Figure 5: *(left)* Benchmark test for evaluation of the approach to detect user intent changes. In the tests, the driver sees the wheelchair and the desired intention from a top-view perspective (shown at bottom, left). *(right)* Adopted user model p_{user}, showing the likelihood of each user signal depending on the relative angle of the goal position w.r.t. the wheelchair.

shows the other extreme, where we want our algorithm to always find an inconsistency. This is the case where p_{pred} has all probability mass focused on one intent, and p_{user} has all probability mass focused on another intent. As a result, $\alpha_{KLD} \leq 22.21$ if $n = 24$. A variation of this is the situa-

tion where p_{pred} has all probability mass equally distributed over h intents, and p_{user} has all probability mass equally distributed over the other $n - h$ intents. For this case, Figure 6 (c) shows the KLD as a function of h. A minimum can be seen for $h = n/2$, for $n = 24$ this is $\alpha_{KLD} = 15.12$. Other scenarios can be thought of, but for our experiments we use this minimum KLD as a threshold for detecting an intent change. Figure 6 (d) shows that the adopted α_{KLD} is dependent on the chosen minimum probability α_{min}.

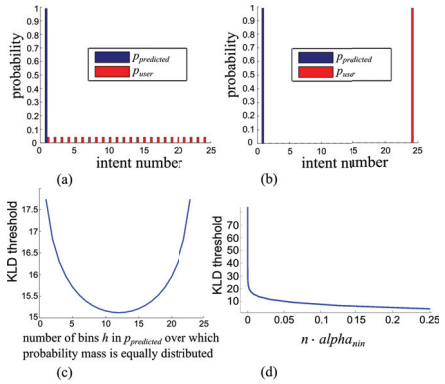

Figure 6: Boundaries for appropriate α_{KLD} values (see text for explanation).

Tests and discussion. Figure 7 shows the trajectories executed by the user in full control (a), in shared control mode with (b) and without (c) KLD consistency test. Figure 7 (d) shows the evolution of the KL distance over time for the shared control approach with KL consistency test (corresponding to (b)). Green circles in (a) indicate positions where an actual intent change occurs. In (b), green squares correspond to correctly detected intent changes, magenta squares to undetected intent changes and red squares to false alarms (an intent change was estimated though there was no intent change at that time instant). Figures (b) and (c) also show the paths from figure (a) in grey, as these grey paths serve as a reference.

These results indicate that the consistency test is indeed effective in detecting intent changes correctly, though it is sometimes delayed by one time step. This intent change detection has a positive effect on the behavior of the shared control assistance. Compared with the user control mode, shared control allows the user to reach his or her goal positions with fewer user signals (352 in user control mode versus 175 in shared control mode with consistency checks). Compared with the shared control mode without KLD consistency checks, the paths in shared control mode with consistency checks deviate much less from the ideal path (which we assume to be similar to the paths in user control mode), and assistance is much more reactive after intent changes.

As a downside of using the consistency check, we have noticed the occurrence of some false alarms, where the intent estimator incorrectly believes the user changed his or her mind. The consequence of this is that the probability distribution is uniformized and that the robot behaves much more "impulsively", taking only the latest user signal into

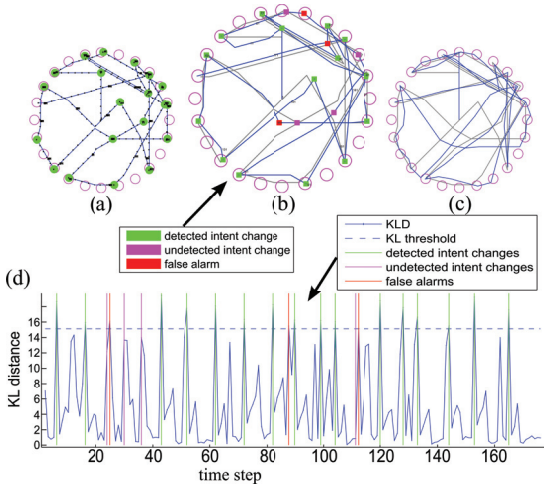

Figure 7: Trajectories executed by the user in full control (a), in shared control mode with (b) and without (c) KLD consistency test. Figure (d) shows the evolution of the KL distance over time for the shared control approach with KL consistency test (corresponding to (b)).

account. This should be avoided, but it cannot be realized by raising the α_{KLD} threshold, because the KL distance at these false alarms was at times higher than the KL distance at actual intent changes. Although further in-depth research is required, the cause of the false alarms seems to lie in the (too) narrowly defined user intention: we have adopted a single straight-line path from current robot pose to the goal position. In reality however, many paths exist to reach that goal position. In this setup, users sometimes prefer to follow a less "optimal" (in the sense of: shortest) path if that suboptimal path would involve less user signal changes, because changing to a different interface button involves some energy that may be experienced as more expensive than reaching the goal location as fast as possible.

Conclusions and Future Work

Our long-term goal is to develop a modular control framework for assistive robots that aid their users in executing *any* task that is possible with the robot in an intuitive way. For robotic wheelchairs, we attempt to realize this by modeling intentions as mental trajectories that the user has in mind from the current robot pose to a desired goal pose. In order to avoid "impulsive" robot behavior, information gathered in the past is accumulated in a probability distribution over user intentions to accurately understand the user's plan. In case the user changes his or her mind however, evidence from the past should (at least temporarily) be discarded. This paper focused on detecting such intent changes using the symmetric Kullback-Leibler distance. We have obtained promising results with this approach in our proof-of-concept tests. Nevertheless, sometimes false alarms are raised. In the near future, we intend to optimize this approach and compare it to other goodness-of-fit tests (Narsky and Porter 2013).

References

Carlson, T., and Millán, J. d. R. 2014. Brain-controlled wheelchairs: A robotic architecture. *IEEE Robotics and Automation Magazine* 20(1):65–73.

Demeester, E.; Huntemann, A.; Vanhooydonck, D.; Vanacker, G.; Van Brussel, H.; and Nuttin, M. 2008. User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving. *Autonomous Robots* 24(2):193–211.

Demeester, E., H. A. V. P. E. D. S. J. 2014. ML, MAP and greedy POMDP shared control: Qualitative comparison of wheelchair navigation assistance for switch interfaces. *Journal of the Chinese Society of Mechanical Engineers* 35(4):333–342.

Fox, D.; Burgard, W.; and Thrun, S. 1998. Active markov localization for mobile robots. *Robotics and Autonomous Systems* 25:195–207.

Fox, D. 2003. Adapting the sample size in particle filters through KLD-sampling. *The International Journal of Robotics Research* 22(12):985–1003.

Hüntemann, A.; Demeester, E.; Vander Poorten, E.; Van Brussel, H.; and De Schutter, J. 2013. Probabilistic approach to recognize local navigation plans by fusing past driving information with a personalized user model. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 4376 – 4383.

Jensfelt, P., and Kristensen, S. 2001. Active global localization for a mobile robot using multiple hypothesis tracking. 17(5):748–760.

Lankenau, A., and Röfer, T. 2001. A versatile and safe mobility assistant. *IEEE Robotics & Automation Magazine* 8(1):29–37.

Larsen, T. D. 1998. *Optimal Fusion of Sensors*. Ph.D. Dissertation, Technical University of Denmark, Department of Automation.

Lenser, S., and Veloso, M. 2000. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of the 2000 IEEE International Conference on Robotics & Automation (ICRA'00)*, 1225–1232.

Montesano, L.; Diaz, M.; Bhaskar, S.; and Minguez, J. 2010. Towards an intelligent wheelchair system for users with cerebral palsy. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18(2):193–202.

Narsky, I., and Porter, F. C. 2013. *Statistical Analysis Techniques in Particle Physics: Fits, Density Estimation and Supervised Learning*. Wiley-VCH.

Parikh, S. P.; Grassi Jr., V.; Kumar, V.; and Okamoto Jr., J. 2004. Incorporating user inputs in motion planning for a smart wheelchair. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation (ICRA)*, volume 2, 2043 – 2048.

Porta, J.; Verbeek, J.; and Kröse, B. 2005. Active appearance-based robot localization using stereo vision. *Autonomous Robots* 18(1):59–80.

Simpson, R. C., and Levine, S. P. 1999. Automatic adaptation in the navchair assistive wheelchair navigation system. *IEEE Transactions on rehabilitation engineering* 7(4):452–463.

Yanco, H. A. 2000. *Shared User-Computer Control of a Robotic Wheelchair System*. Ph.D. Dissertation, Massachusetts Institute of Technology MIT, Department of Electrical Engineering and Computer Science.