Autonomous Shopping Cart Platform for People with Mobility Impairments

L. Marchetti, D. Pucci,, Pascal Morin

Autonomous Shopping Cart Platform for People with Mobility Impairments

Luca Marchetti[1] and Daniele Pucci[1] and Pascal Morin[2]

***Abstract*— Providing a platform able to interact with a specific user is a challenging problem for assistance technologies. Among the many platforms accomplishing this task, we address the problem of designing an autonomous shopping cart. We assume that the shopping cart is set-up on a unicycle-like robot endowed with two sensors: an RGB-D camera and a planar laser range finder. To combine the information from these two sensors, a data fusion algorithm has been developed using a particle filter, augmented with a k-clustering step to extract person estimations. The problem of stabilizing the robot's position at a fixed distance from the user has been solved through classical control design. Results on a real mobile platform verify the effectiveness of the approach here proposed.**

I. INTRODUCTION

Assistive technologies focus their efforts on providing reliable solutions to help people in the everyday life. One of the key components of an assistive system is the ability to actively follow a user, a task well exemplified by a mobile robot that follows the user. An *autonomous shopping cart* is a simple application that provides a good test-bed for a whole class of problem: a robotic butler that helps on carrying heavy objects; a robotic lift that has to follow a companion to accomplish a coordinated task; an automatic walking aid that should support elderly people and so on.

Platform of such kind should be able to detect and recognize the user, among other people, and be capable of following continuously the same user. The environment should be modelled in such a way the robot can avoid obstacles, and pursue the user at the same time. We focused our attention on developing methodologies to accomplish a safe following of user's trajectory, while maintaining a certain degree of freedom on the reference position of the robot w.r.t. the user.

The involved scientific challenges can be summarised in two aspects. On one hand, a module must be developed in order to estimate the user position while identifying other people in the environment. The selection of the user should be effective in such a way the continuous following of the user will not be confused by the presence of other subjects. This estimation must achieved in a cluttered and noisy environment. On the other hand, one has to provide a reliable and feasible way to control the mobile platform, respecting the peculiarity of human motion and exploiting the robotic aid.

[1]INRIA Sophia Antipolis Meditérranée, 2004, Route des Lucioles - BP 93 Sophia Antipolis Cedex 06902 <name>.<surname>@inria.fr

[2]ISIR-UPMC, 4, place Jussieu 75005 Paris <surname>@isir.upmc.fr

Combining these two challenges represents the main contribution of our work. *People Position Estimation* is a well-known problem addressed in many scenarios, e.g. video-surveillance [1] or activity recognition [2]. In the field of object recognition, the human body represents probably the most challenging one. The complexity of shape, as well as the multiplicity of configuration it can achieve require complex sensors to be captured. Color cameras are amongst the most effective sensors, even if the information are limited to the image plane [3]. Recently the evolution of technology and availability of relatively cheap RGB-D sensors, capable of perceiving 3D structures, opened the possibility to extend the range image-based recognition [4]. It is then reasonable to choose such sensors to capture people positions[5].

These devices, however, usually have a limited field-of-view. While it is completely reasonable to use multiple camera to augment the virtual field-of-view [6], other aspects of the application guided us on choosing a different solution to cope with this problem. In the case of a robot following a person, it is important to take care of obstacles that can limit the motion of the robot. To address this problem, usually a *laser range finder* is employed to map the surrounding [7]. Its high precision on a 2D plane is an effective way to detect obstacles for the robotic motion. Moreover, the laser information can also be used to detect and track the legs of several people [8]. Thanks to the large field of view and range, and the sensor resolution, the user position can be estimated with high accuracy.

By exploiting the strengths of both sensors, an improved position estimation can be obtained. The result is an accurate person position estimation that can be given as input to a person following control module. While the human being is able to move along any direction, a wheeled robotic platform is usually subjected to kinematic constraints that limit the range of feasible trajectories [9]. Hence, to achieve human following by the robot, additional maneuvers may be necessary when the human trajectory does not satisfy the aforementioned constraints. This problem has been largely studied in the last decades and one can rely on existing techniques to address the control problem [9].

In this paper, we describe how a data fusion algorithm, that combines information from 2D and 3D sensors, can be effectively coupled with a trajectory stabilization method, able to drive the robot to solve the person following challenge.

II. NOTATION

We consider the class of unicycle-like robots sketched in Figure 1. The following notation is used. Let $\mathcal{I} = \{O; \vec{\imath}_0, \vec{\jmath}_0\}$

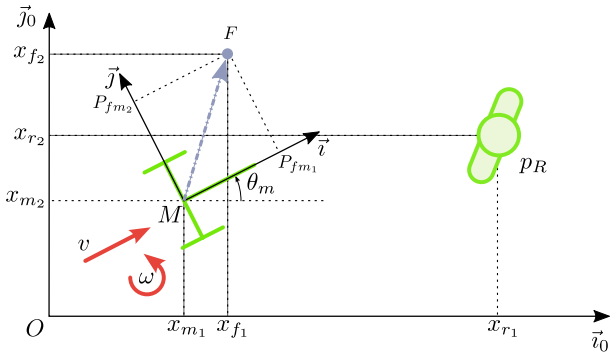

Fig. 1. Unicycle-like robotic platform for autonomous shopping cart.

be a fixed inertial frame with respect to (w.r.t.) which the robot's absolute pose is measured. The point M is the middle point of the wheel's axis, and $\mathcal{B} = \{M; \vec{\imath}, \vec{\jmath}\}$ is a frame attached to the robot. The vector $\vec{\imath}$ is perpendicular to the wheel's axis. The vector of coordinates of M in the basis of the fixed frame $\mathcal{I}$ is denoted as $x_m = (x_{m_1}, x_{m_2})^T$. Therefore, $\vec{OM} = x_{m_1}\vec{\imath}_0 + x_{m_2}\vec{\jmath}_0$. The robot's orientation is characterized by the angle θ_m between $\vec{\imath}_0$ and $\vec{\imath}$. The rotation matrix of an angle θ_m in the plane is $R(\theta_m)$. With $\{e_1, e_2\}$ we denote the canonical basis in $\mathbb{R}^2$. In view of this notation, the kinematic model of the robot writes [9]

$$\begin{aligned} \dot{x}_m &= v R(\theta_m) e_1 \\ \dot{\theta}_m &= \omega. \end{aligned} \tag{1}$$

with v the robot's rolling velocity and ω its rotational velocity considered as kinematic control inputs. The position of the user is represented by a *reference point* p_R. The vector of coordinates of p_R in the basis of the fixed frame $\mathcal{I}$ is denoted as $x_r = (x_{r_1}, x_{r_2})^T$. Therefore, $\vec{Op}_R = x_{r_1}\vec{\imath}_0 + x_{r_2}\vec{\jmath}_0$. The vector of coordinates associated with the linear velocity of p_R w.r.t. $\mathcal{I}$ is denoted as $\dot{x}_r$.

III. People Position Estimation

People detection is one of the main component needed to have a reliable autonomous shopping cart. In Section I, we highlighted the advantages of using two different sensors to achieve the position estimation. Next, we describe the data fusion architecture pointing out the main characteristics. The functional blocks are described in Figure 2.

The architecture is two-tiered. The lowest level has a person estimation method for each sensors. The highest level combines the two estimations to obtain a more reliable position estimation.

a) Laser-based position estimation: To detect people using the laser range finder, we use an implementation of the *Kalman Filter leg tracker*, described by Arras et al in [8]. Therefore, we follow the notation presented in that paper to briefly describe the functioning of this estimator.

The *KF-based multi-hypotheses tracker* describes a leg track as $p_L^i = (p_x, p_y, v_x^p, v_y^p)$, with p_x and p_y position on the laser plane, and v_x^p and v_y^p the components of the velocity. The *state prediction* of the leg filter uses a constant-velocity model. The *observations* of legs are detected using an Adaboost algorithm [10] that classifies the segments found in the laser scan according to a set of features. Using these features, several weak classifiers are used to separate leg candidates. The combination of all the classifiers generates the leg observations. The training procedure and how to obtain a valid set of informative threshold to classify the segments is explained in [11]. The tracker labels assign each new measurement to existing leg tracks or creates new tracks. At any instant, a leg track can be *detected* (if measurements are assigned during the last observation phase) or *deleted* (if measurements are not assigned). New tracks are labeled as *new track* and a *false alarm* label is used when the measurements are mistakenly detected as track.

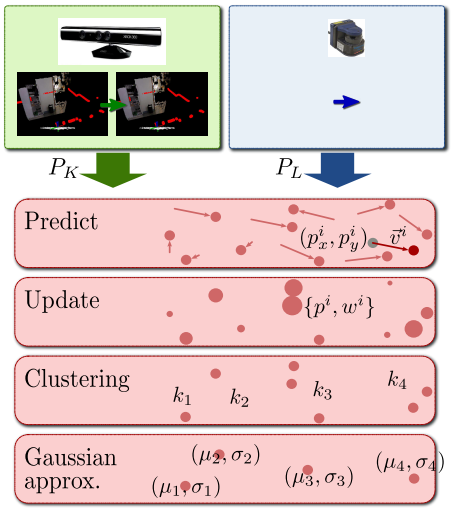

Fig. 2. The data fusion architecture for person position estimation.

People tracks are extracted from leg tracks using the following heuristic:

- people have two legs;
- legs are close to each other;
- legs move in similar direction;
- legs have a higher probability of occluding each other, than being occluded by other people's legs or objects.

This model is implemented in such a way it takes into account the possible occlusions, thus avoiding deletion of track if legs are occluded for a short period of time. Other considerations about leg track labeling and probability association are described in details in [8] and will be omitted here.

b) Kinect-based person estimation: The availability of a ROS-integrated library for user skeleton detection simplifies the problem of detecting human shapes using the RGB-D data from the Kinect sensor. As for OpenNI library version 1.3, the user needed to perform a peculiar calibration procedure (the so-called ψ pose) to be detected. This limited the usability of the bundle software for multiple people detection. As for version 1.5, however, the users can be detected using a *User Generator* that does not require any calibration at

all. The output of this module is a set of people position estimation P_K and can be extremely noisy, because the people's bodies are recognized applying statistical methods. This required to develop appropriate filtering procedure to establish correct person following.

c) Data-fusion for person position estimation: We decided then to use both information, from the laser and the RGB-D camera, to provide more reliable information to the control layer. The camera can be really accurate on estimating the complexity of human body, while the laser provides a larger field of view and a better precision on distance estimation.

To combine the advantages of both sensors, we designed a *particle filter* with *clustering*. Each particle represents a possible position estimation as $p^i = (p_x, p_y, v_x^p, v_y^p)$. The posterior density is approximated by:

$$\mathcal{P}(X|z) \approx \sum_{i=1}^{|P|} w^i \delta(X - x^i), \tag{2}$$

where the X is the current state of the probability density function and w^i is a weight associated to the sample $x^i \in X$. The interested reader can find more explanation about this representation in [12].

The algorithm is described below.

Algorithm 1: People Position Estimation

Data: $P_L :=$ {position estimation from laser}
$P_K :=$ {position estimation from Kinect}
$P :=$ {particle set}

1 **for** $p^i \in P$ **do**
2 draw particles: $\widetilde{p}_t^i \sim \pi_t(p^i|P, \vec{v}_{t-1}^i)$
3 calculate weight: $\widetilde{w}_t^i \propto \mathcal{L}_L(\widetilde{p}_t^i) \times \mathcal{L}_K(\widetilde{p}_t^i)$
4 resample: $\{p_t^i, w_t^i\}_{i=1}^{|P|} = resample(\{\widetilde{p}_t^i, \widetilde{w}_t^i\}_{i=1}^{|P|})$
5 get position estimation clusters: $\mathcal{C} = KClusterise(P)$

At time t, the particle filter algorithm requires a proposal distribution (π_t) from which it draws samples during the *prediction step*. We use the previous set of particles evolved using a constant velocity model. The *update step* uses information from the laser and the RGB-D sensors. A sensor model calculate the likelihood of each particle to belongs to the set of laser measurements P_L or camera measurements P_K. The likelihood is evaluated as:

$$\mathcal{L}_L = \frac{prob(P_L|p_t^i)prob(\widetilde{p}_t^i|p_{t-1}^i)}{\pi(\widetilde{p}_t^i|\widetilde{P}, \vec{v}_{t-1}^i)}, \tag{3}$$

$$\mathcal{L}_K = \frac{prob(P_K|p_t^i)prob(\widetilde{p}_t^i|p_{t-1}^i)}{\pi(\widetilde{p}_t^i|\widetilde{P}, \vec{v}_{t-1}^i)}. \tag{4}$$

The estimated posterior represents the distribution of people over the sensor's space. This posterior is usually multimodal, given the noisy nature of the RGB-D camera and ambiguity on laser estimation. Therefore, a clustering phase is necessary to extract all the possible tracks. A track, or a person estimation, will be the Gaussian approximation (mean and variance) of a single cluster.

A selection procedure, not described here, selects the best candidate and assigns it to the control module.

A. K-Clustering

We implemented a *k-clustering* based technique[13], described in Algorithm 2. *KClusterise* tries to detect up to N_k clusters. Therefore, it is not a free-cluster algorithm, and this could potentially lead to a limitation. However, for the purpose of the presented applications, this is not a critical problem.

Algorithm 2: KClusterise

Data: P: particle set
K: cluster set, of maximum size N_K
O: outliers set

1 Initialise cluster set: $K \leftarrow \varnothing$
2 Find cluster: $K = FindCluster(P)$
3 Assign particle to cluster:
$O = ClassifyParticles(K, P)$
4 Redistribute outliers: $SpreadOutliers(O, K)$

Algorithm 3: FindCluster

```
// Find equally spaced out centroids
```
1 **for** $p \in P$ **do**
2 *isFar* = true
3 **for** $k \in K$ **and** *isFar* **do**
4 *isFar* $= (\|p, k\| > \delta_{far})$
5 **if** *isFar* = *true* **then**
6 Add particle p as centroid: $K \leftarrow p$
7 **return** K

First, the algorithm tries to find N_k points (with $N_k \geq 1$) that are equally spaced out (Algorithm 3). Adding particles in line 6 is done using a priority queue principle, considering the distance. At the end of procedure, K will contain the most distant N_K points, and they will be used as centroids.

Successively (Algorithm 4), for each point p, it calculates the Euclidean distance δ between p and clusters $k \in K$. Let δ_{min} be the minimum distance between p and a cluster k.

Two cases are possible: if δ_{min} is less than the *threshold distance* δ_{far} (discussed in Section III-B), p will be put in cluster k. Otherwise, p will be put into the *outliers set* O.

When the clusterisation phase is finished, the points in the outliers set O are evaluated. Each point in O will be put in the nearest cluster by ignoring the threshold distance.

B. Threshold Distance Function

One of the major problems in clustering techniques is to find a threshold distance to approximate the correct number of clusters. There are two possible ways: a fixed value

Algorithm 4: ClassifyParticles

```
// Classify particles
for p ∈ P do
    added = false
    for k ∈ K and not added do
        δ_min = ||p, k||
        if δ_min < δ_far then
            Add particle p to cluster k: k ← p
            added = true
    if added = false then
        Add particle p to outliers set: O ← p
return O
```

Algorithm 5: SpreadOutliers

```
// Redistribute outlier particles
for p ∈ O do
    δ_min = ∞
    for k ∈ K do
        δ = ||p, k||
        if δ < δ_min then
            k_candidate = k
            δ_min = δ
    Classify particle p: k_candidate ← p
```

or a variable one. The first choice can be computationally efficient, but it is not flexible w.r.t. environment's changes.

We adopted the second strategy, by using a *dynamic threshold function*. For each position estimation in P_L, P_K, we evaluate the average distance to other estimations as:

$$\delta_{far} = \frac{\sum_{i=1,j=2}^{|P_L|} \|p_L^i, p_L^j\| + \sum_{i=1,j=2}^{|P_K|} \|p_K^i, p_K^j\|}{|P_L| + |P_K|}, \quad i \neq j. \quad (5)$$

The idea behind this function is to keep the position estimation as far as possible to each other. Using the average of pairwise distance helps us to obtain well-balanced cluster while keeping them separated.

IV. Control Design

Achieving a reliable person following requires correct control laws to smoothly let the robot follow a trajectory constrained by the person's motion. Our objective is to describe the desired position of the person w.r.t. the robot, then minimize the distance between this desired position and the actual position. From Figure 1 the position of the desired (or *follower*) point F is given by $x_f = x_m + R(\theta_m)P_{fm}$, where P_{fm} is the vector of coordinates of $\vec{MF} = \vec{OF} - \vec{OM}$. Recall that the position of the person, is given by p_R. This is the results of person position estimation described in Section III.

Let $\tilde{x}$ be the position error in the fixed frame, defined by:

$$\tilde{x} = x_f - x_r, \quad (6)$$

and w.r.t. the mobile frame by:

$$\tilde{p} = R(\theta_m)^T \tilde{x}. \quad (7)$$

Note that here $x_r = (p_x, p_y)$. Therefore the error dynamics w.r.t. the mobile frame writes:

$$\dot{\tilde{p}} = -\omega S\tilde{p} + Mu - R(\theta_m)^T \dot{x}_r(t), \quad (8)$$

with $M = \begin{bmatrix} 1 & -P_{fm_2} \\ 0 & P_{fm_1} \end{bmatrix}$, $S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, $u = (v, \omega)^T$ and $\dot{x}_r = (v_x^p, v_y^p)^T$. Relying on the results in [9], one deduces the following lemma.

Lemma 1: Assume that $P_{fm_1} \neq 0$, so that $det(M) \neq 0$. Apply the control input

$$u = -M^{-1}\left[K\tilde{p} - R(\theta_m)^T \dot{x}_r(t)\right], \quad (9)$$

where $K = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}, K > 0$. Then,

$$\dot{\tilde{p}} = -\omega S\tilde{p} - K\tilde{p}, \quad (10)$$

so that $\tilde{P} = 0$ is a globally asymptotically stable equilibrium point for the closed-loop system.

The stability analysis follows by verifying that $V(\tilde{p}) = \|\tilde{p}\|^2$ is a strict Lyapunov function for the closed-loop system. When $P_{fm_1} = 0$, the control law (9) is not well defined and other solutions to the tracking problem must be considered. This problem will be addressed in a forthcoming paper.

V. Results

The first implementations of the mobile platform have been conducted on an industrial wheeled robot. The robot (cfr. Figure 3) has a Kinect mounted on a pan/tilt unit, which allows a visual servoing procedure to follow a user moving all around the robot. The OpenNI[1] library is used to detect and estimate the raw positions of *all* people in the field-of-view of the camera. A laser is mounted on the front side of the robot and captures scans of the environment. This scans are used to estimate the odometry of the robot (using the *Canonical Scan Matcher*[2]). The onboard computer manages the visual servoing routine, controlling the pan/tilt unit. The attached laptop, instead, processes data from Kinect and laser, to model the pose of people and evaluate control inputs.

The software architecture has been developed using the ROS[3] framework. The actual architecture, illustrated in Figure 4 is composed of three layers: the *robot interface*, the *modeling* core, and the *behaviour* component.

The *robot interface* provides the abstraction layer between the actual platform and the software components. It is easy to adapt this interface for several platforms and keep unchanged the higher levels.

[1] http://www.openni.org
[2] http://ros.org/wiki/csm
[3] http://www.ros.org

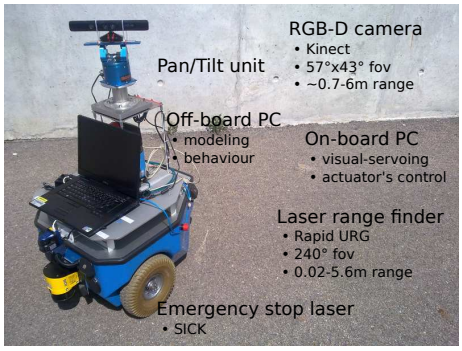

Fig. 3. The robot used to run the experiments.

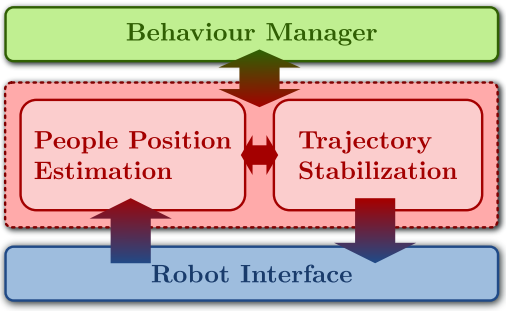

Fig. 4. The software architecture and modules developed for the autonomous shopping cart robot.

The *modeling* layer detects the user within the sights of the sensors and evaluates the control inputs. The user position estimation is evaluated as described in Section III, while the control inputs are the results of control laws developed in Section IV.

The last layer, the *behaviour* component, handles the safety measures to ensure the robots avoid close obstacles (detected by the laser), starts and stops services on request, enables initialization procedures and so on. In particular, it selects the first user to be followed, among possible candidates.

A. Experimental Results

In this section, we focus on the practical experiments obtained using the robot. We present three different configuration, considering the behaviour and estimation performed by the robot when the follower point x_f was placed in two different positions: $A = (1.5, 0)$ and $B = (1, 1)$. The positions are depicted in Figure V-A. Supplement material and high quality versions of the picture presented here can be found at `http://goo.gl/NFnjg`.

1) Test A: user in front of the robot: Figure 6 presents the outcome of the data fusion procedure. The data-fusion trajectory represents the trajectory of the person passed to the control module. Despite the high number of hypotheses, the data fusion algorithm is able to consistently track the movement of the person. In Figure 7 is presented the trajectory of the following point w.r.t. the person estimation: the control inputs are consistent with the estimation. The

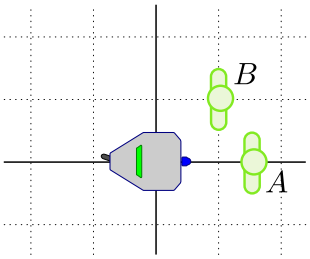

Fig. 5. The position of the follower point x_f used during tests.

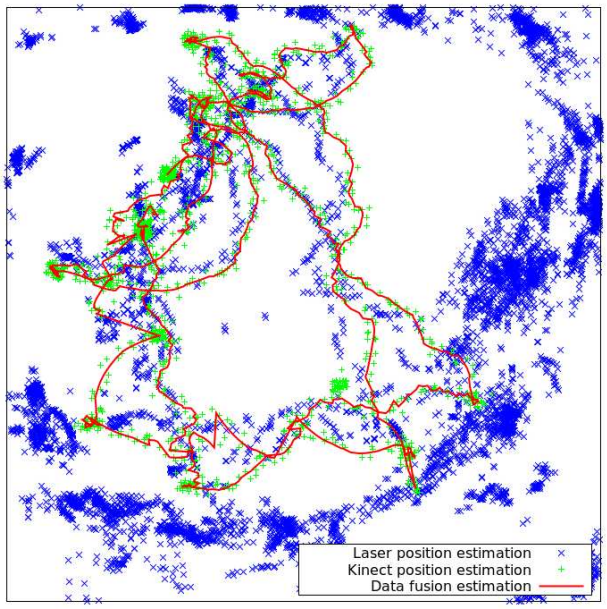

Fig. 6. Results for the data fusion algorithm during test A: $x_f = (1.5, 0)$.

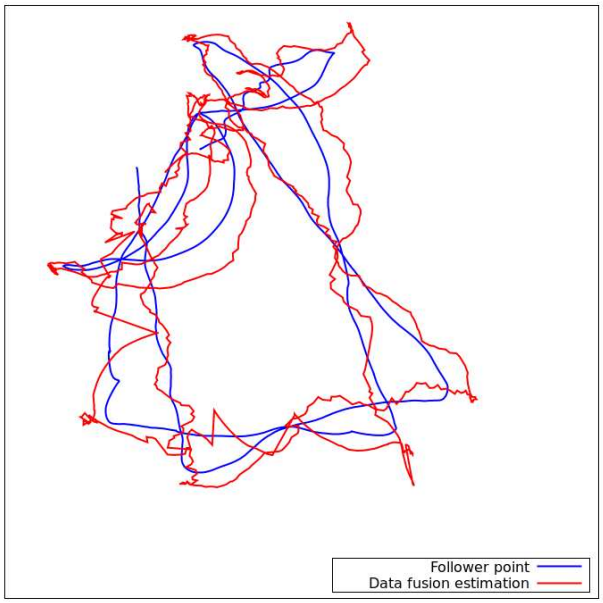

Fig. 7. Trajectory of robot given the control input w.r.t. the position estimation for test A $x_f = (1.5, 0)$.

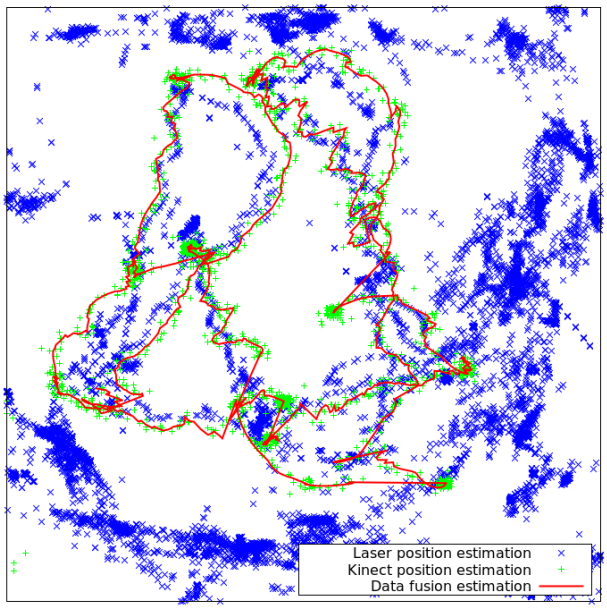

Fig. 8. Results for the data fusion algorithm during test B: $x_f = (1, 1)$.

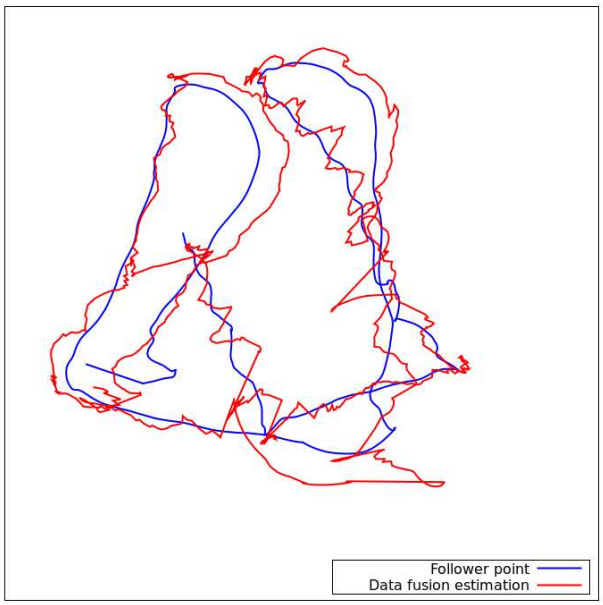

Fig. 9. Trajectory of robot given the control input w.r.t. the position estimation for test B $x_f = (1, 1)$.

movement of the robot resulted much more smoothed than the estimation. This effect is due to the different frequency at which the control commands are sent to the motor (much lower than the estimation rate).

2) Test B: user at $45°$ *in front of the robot:* In Figure 8 are presented the same results as for test A, showing the effectiveness of the data fusion algorithm. In Figure 9 are presented the trajectory of the following point while at $45°$ w.r.t. the robot. The results are consistent with the test A.

VI. Conclusions

This article presented a reliable solution to the problem of person following. A data fusion algorithm has been presented to reliably detect and estimate the position of multiple people. Two kinds of sensors have been exploited to accomplish the position estimation: a laser range finder and a RGB-D camera. Based on this estimation, a feedback control law has been designed to track the user. To demonstrate the effectiveness of the proposed architecture, results have been presented using a real mobile platform. As future work, we are investigating how to address the singularity on the control design. A different control law needs to be developed to avoid the singularity, in order to have a completely arbitrary position for the following point x_f.

Acknowledgements

We would thank the Large Scale Initiative Action PAL (*Personally Assisted Living*), that funded this ongoing research, and the people working at INRIA Sophia Antipolis Meditérranée that help on pursuing this work.

References

[1] L. M. Fuentes and S. A. Velastin, "People tracking in surveillance applications," in *In Proceedings of the 2nd IEEE International workshop on PETS*, 2001.

[2] R. Bodor, B. Jackson, N. Papanikolopoulos, and H. Tracking, "Vision-based human tracking and activity recognition," in *Proc. of the 11th Mediterranean Conf. on Control and Automation*. Kostrzewa Joseph, 2003, pp. 18–20.

[3] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people." *Computer Vision and Image Understanding*, vol. 80, no. 1, pp. 42–56, 2000.

[4] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images," in *Computer Vision and Pattern Recognition*, June 2011.

[5] L. Spinello and K. O. Arras, "People detection in RGB-D data," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, San Francisco, USA, 2011.

[6] M. Luber, L. Spinello, and K. O. Arras, "People tracking in rgb-d data with on-line boosted target models." in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[7] A. Censi, "An ICP variant using a point-to-line metric," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[8] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, USA, 2008.

[9] P. Morin and C. Samson, *Handbook of Robotics*. Springer, 2008, ch. Motion control of wheeled mobile robots, pp. 799–826.

[10] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, ser. EuroCOLT '95. London, UK, UK: Springer-Verlag, 1995, pp. 23–37.

[11] K. O. Arras, scar Martnez, and M. W. Burgard, "Using boosted features for detection of people in 2d range scans," in *In Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2007.

[12] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2001.

[13] A. K. Jain and R. C. Dubes, "Algorithms for Clustering Data," in *Prentice Hall, Englewood Cliffs New Jersey*, 1988.