Interactive Mapping Using Range Sensor Data under Localization Uncertainty

Received 10th October 2012; accepted 22nd November 2012.

Pedro Vieira, Rodrigo Ventura

Abstract:
Several methods have been proposed in the literature to address the problem of automatic mapping by a robot using range scan data, under localization uncertainty. Most scan matching methods rely on the minimization of the matching error among individual range scans. However, uncertainty in sensor data often leads to erroneous matching, hard to cope with in a purely automatic approach. This paper proposes a semi-automatic approach, denoted interactive mapping, involving a human operator in the process of detecting and correcting erroneous matches. Instead of allowing the operator complete freedom in correcting the matching in a frame by frame basis, the proposed method facilitates the adjustment along the directions with more ambiguity, while constraining the others. Experimental results using LIDAR data are presented to validate empirically the approach, together with a preliminary user study to evaluate the benefits of the approach.

Keywords: *2D mapping, interactive interface, ICP, geometric constraint analysis, iterative methods*

1. Introduction

The problem of automatic environment mapping by a robot has been an active field of research for a long time (see [1] for a review). Methods vary both in terms of sensors used (*e.g.*, sonars [2], LIDAR [3], vision [4], and more recently the Kinect [5], [9]), and in methodologies (*e.g.*, probabilistic [6], scan matching [3]). However, most of these methods are prone to local minima, originated, for instance, by ambiguity or by locally periodic patterns in the environment. Moreover, the loop closure problem remains a not completely solved problem [7]. Approaches aiming at global consistency [3] have been proposed, however, they are often computationally complex, and yet prone to error when faced with missing data.

In this paper we propose an alternative approach where we consider the human-in-the-loop of the scan matching process. In particular, the user is asked to interact with the matching process, by adjusting the match of individual pairs of scans. This adjustment is however constrained by favouring adjustments along the directions of greater ambiguity. Take for instance a case of pairs of identical scans taken from a homogeneous corridor (Fig. 1): the system favours the adjustment of the scans along the corridor, while disfavouring movements orthogonal to the corridor.

The proposed method is based on a graphical user interface (GUI), where the user interacts with the system using a common computer interface (mouse and keyboard). Consider a pair of range scans denoted M (for model) and D (for data). Fig. 1 illustrates the situation for the corridor example. When the user drags one of the range scans, say

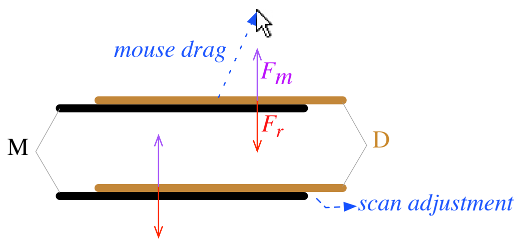

Fig. 1. Two corridors being align. The scan D is adjusted such that the reaction force $\mathbf{F_r}$*, computed from the cost function gradient, balances the force* $\mathbf{F_m}$ *imposed by the mouse drag.*

scan D, using the mouse, a corresponding virtual force $\mathbf{F_m}$ is produced (proportional to the drag vector). Opposing it, a reaction force $\mathbf{F_r}$ is computed based in the match between scans M and D. Considering the scan matching cost function as a potential function, the symmetric of its gradient with respect to the shift of scan D corresponds to the reaction force $\mathbf{F_r}$. This reaction force "attracts" scan D towards M, along the direction of less ambiguity. Scan D is then moved such that both forces become balanced, $\mathbf{F_m} + \mathbf{F_r} = 0$. In the corridor example in Fig. 1, the reaction force is (mostly) orthogonal to the corridor axis.

Although the method has been implemented for 2D mapping, it can be easily extended to 3D. On the one hand, the matching cost function is applicable to 3D, and on the other, 2D mouse movements can be mapped into 3D forces, depending on the map viewpoint in the GUI, for instance. The work presented here is a preliminary study towards a full 3D implementation.

Such method can be used to make accurate maps, that can then be used, for instance, in search and rescue missions. This missions can vary from searching for victims within risky areas, to analysing certain areas for better mission planning. The availability of a map of these areas can improve dramatically the efficiency and effectiveness of these operations.

Little related work can be found in the literature concerning interactive mapping. There is however some work on the interactive integration of semantic information in maps. Systems which have explored this topic include SemanticSLAM [10] and Interactive SLAM [11], both prototypes not developed beyond their early stages.

This paper is organized as follows. In Section 2 the method for interactive mapping is proposed, followed by Section 3 showing the results, together with a small user study. Finally, Section 4 draws some conclusions and discuss possible future work directions.

2. Interactive Alignment

Consider two scans in the 2D space, $M = \{\mathbf{m_k}\}$ and $D = \{\mathbf{d_k}\}$, for $\mathbf{m_k}, \mathbf{d_k} \in \mathbb{R}^2$, and an initial rigid transformation (R,t) which align D with M, where R is a rotation matrix parametrized by an angle θ:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \qquad (1)$$

and $\mathbf{t}$ is a translation vector. Although this transformation can be initialized with a default value (*e.g.*, $R|_{\theta=0} = I_{2\times 2}$ and $\mathbf{t} = (0,0)^\top$), we start by using the Iterative Closest Points (ICP) algorithm (see [8]) to get the best initial transformation automatically. By applying this transformation to D, a transformed scan, D', is obtained. The two scans (M and D') are then presented in a viewer (GUI) for alignment analysis. If the alignment isn't well done, the user interacts with the viewer using common computer mouse and keyboard, and apply either translations or rotations to D', in order to correct it. These actions are carried out separately using a designated key to choose which mode to use.

Scans are aligned by balancing a virtual force originated by a mouse drag, henceforth called *mouse force*. This drag is defined by two points: $\mathbf{p_o}$, the mouse pointer position where the user clicked on the mouse, and $\mathbf{p_f}$, the current pointer position. We consider the mouse force to be proportional to the difference between $\mathbf{p_f}$, and the initial point $\mathbf{p_o}$ transformed by the alignment under way. In the general case, we assume D' rotated by θ with respect to a center of mass $\mathbf{c}$, and then translated by $\mathbf{t}$. Then, point $\mathbf{p_o}$ is transformed into $\mathbf{p_o}' = R(\theta)(\mathbf{p_o} - \mathbf{c}) + \mathbf{c} + \mathbf{t}$. Now, we define a potential function J_m that grows with the distance between points $\mathbf{p_o}'$ and $\mathbf{p_f}$:

$$J_m = \frac{1}{2}\|\mathbf{p_f} - \mathbf{p_o}'\|^2. \qquad (2)$$

By taking the gradient of this potential, with respect to either translation $\mathbf{t}$ and rotation R, one obtains the virtual forces and torques induced by the mouse drag.

The potential function that minimizes the distances between corresponding points of the two point clouds, and is responsible for creating the reaction force to the mouse drag, is given by:

$$J_r = \frac{1}{2}\sum_{k=1}^{N} \|\mathbf{m_k} - [R(\theta)(\mathbf{d_k} - \mathbf{c}) + \mathbf{c} + \mathbf{t}\|^2, \qquad (3)$$

where $\{\mathbf{m_k}\}$ and $\{\mathbf{d_k}\}$ are pairs of closest points from M and D' respectively, and N is the number of these pairs. This function is similar to the cost function used in ICP. The closest points are computed in the same fashion as in ICP (*e.g.*, using Kd-trees), and only the pairs sufficiently close are considered.

2.1. Translations

In this mode, the alignment consists in a translation by $\mathbf{t}$. Thus, $R(\theta)$ is the identity ($\theta = 0$), and the potential function (2) is simplified. The mouse force $\mathbf{F_m}$ is computed from the gradient of the cost function (2) with respect to the translation $\mathbf{t}$ and evaluated at $\theta = 0$:

$$\begin{aligned} \mathbf{F_m} &= -k_m \nabla_\mathbf{t} J_m|_{\theta=0} \\ &= k_m(\mathbf{p_f} - \mathbf{p_o} - \mathbf{t}), \end{aligned} \qquad (4)$$

where k_m is the proportionality constant. Opposing this force, a reaction force $\mathbf{F_r}$ is computed from the gradient of the cost function (3) with respect to the translation $\mathbf{t}$, and evaluated at $\theta = 0$:

$$\begin{aligned} \mathbf{F_r} &= -k_r \nabla_\mathbf{t} J_r|_{\theta=0} \\ &= k_r \sum_{k=1}^{N} [\mathbf{m_k} - (\mathbf{d_k} + \mathbf{t})], \end{aligned} \qquad (5)$$

where k_r is the proportionality constant. Fig. 2 illustrates the forces involved.

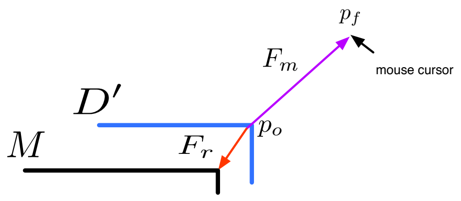

Fig. 2. Geometry of the force balance for translations before alignment ($\mathbf{t} = 0$).

To find a scan adjustment that balances the mouse and the reaction forces, translations are iteratively performed, since each time scan D' moves, the correspondences among points may change. So, for each iteration, a translation $\mathbf{t}$ is computed by solving the equation

$$\mathbf{F_m} + \mathbf{F_r} = 0, \qquad (6)$$

with respect to $\mathbf{t}$. This equation has the following closed form solution:

$$\mathbf{t} = \frac{k_m(\mathbf{p_f} - \mathbf{p_o}) + k_r \sum_{k=1}^{N}(\mathbf{m_k} - \mathbf{d_k})}{k_m + N k_r}. \qquad (7)$$

The scan adjustment results from the algorithm 1.

Algorithm 1 Compute Translation.

```
function TRANSLATION($p_o, p_f, M, D'$)
    $D_{new} \leftarrow D'$
    repeat
        $P \leftarrow ComputeClosestPoints(M, D_{new})$
        $t \leftarrow ComputeTranslation(p_o, p_f, P)$
        $D_{new} \leftarrow UpdateScan(D', t)$
    until $ErrorChange(t) < \xi_t$
    return $t$
end function
```

The algorithm receives, as input, the mouse initial and current position and the two scans being adjusted (M and D'). The algorithm starts by computing a subset, P, of pairs of closest points (*ComputeClosestPoints*) from the two scans ($P \subset [\mathbf{m_k}, \mathbf{d_k}]$). The subset P is then used together with the mouse positions in (7) to compute the translation (*ComputeTranslation*). Finally scan D' is updated (*UpdateScan*) with the new estimate of the translation and the convergence is checked (*ErrorChange*). The algorithm iterates until the correspondences between points are the same, or in other words until the norm of the difference between the new and the previous translation estimation falls below a threshold:

$$\|\mathbf{t_i} - \mathbf{t_{i-1}}\| < \xi_t, \tag{8}$$

where $\mathbf{t_{i-1}}$ and $\mathbf{t_i}$ are the estimation of the translation in iteration $i-1$ and i respectively and ξ_t is a threshold.

The obtained $\mathbf{t} = [t_x\ t_y]^T$ corresponds to the homogeneous transformation:

$$T_t = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{9}$$

2.2. Rotations

Rotation mode comprises a rotation of scan D', with respect to the center of mass $\mathbf{c}$, by an angle θ (Fig. 3). The center of mass of a scan is set to its centroid.

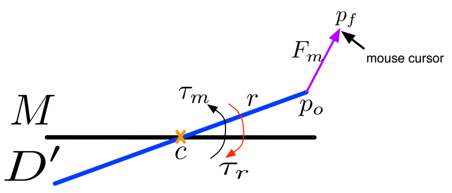

Fig. 3. Torque produced by force originated from a mouse drag prior to alignment ($\theta = 0$).

When the user clicks on the scan D' and attempts to drag it, a force $\mathbf{F_m}$ is created using (2), for $\mathbf{t} = 0$. However, unlike translations, the balance is formulated here in terms of virtual torques. The mouse torque τ_m is obtained by computing the gradient of the cost function (2) with respect to θ, and evaluated at $\mathbf{t} = 0$:

$$\tau_m = -k_m \nabla_\theta J_m|_{\mathbf{t}=0}. \tag{10}$$

The opposing torque τ_r is obtained by computing the gradient of the cost function (3) with respect to θ, and evaluated at $\mathbf{t} = 0$:

$$\tau_r = -k_r \nabla_\theta J_r|_{\mathbf{t}=0}. \tag{11}$$

Both gradients can be computed using the chain rule:

$$\nabla_\theta J_m|_{\mathbf{t}=0} = tr\left[(d_m)^T d_R\right], \tag{12}$$

$$\nabla_\theta J_r|_{\mathbf{t}=0} = tr\left[(d_r)^T d_R\right]. \tag{13}$$

where

$$d_m = \frac{\partial J_m}{\partial R(\theta)}, \tag{14}$$

$$d_r = \frac{\partial J_r}{\partial R(\theta)}, \tag{15}$$

$$d_R = \frac{\partial R(\theta)}{\partial \theta}. \tag{16}$$

The partial derivative (16) can be computed from the derivative of (1) with respect to θ:

$$d_R = \frac{\partial R(\theta)}{\partial \theta} = AR(\theta), \tag{17}$$

where $A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

The partial derivatives (14) and (15) are trivially obtained by computing the derivative of (2) and (3), respectively, with respect to R:

$$d_m = -\mathbf{p_f}'\mathbf{r_i}^T, \tag{18}$$

$$d_r = -\sum_{k=1}^{N} \mathbf{m_k}'(\mathbf{d_k}')^T, \tag{19}$$

where $\mathbf{r_i} = (\mathbf{p_o} - \mathbf{c})$, $\mathbf{p_f}' = (\mathbf{p_f} - \mathbf{c})$, $\mathbf{d_k}' = (\mathbf{d_k} - \mathbf{c})$ and $\mathbf{m_k}' = (\mathbf{m_k} - \mathbf{c})$.

Both torques are obtained by computing the trace of (12) and (13) and using the partial derivatives (17), (18) and (19):

$$\tau_m = k_m\, \mathbf{r_i}^T R^T A^T \mathbf{p_f}', \tag{20}$$

$$\tau_r = k_r \sum_{k=1}^{N} (\mathbf{d_k}')^T R^T A^T \mathbf{m_k}', \tag{21}$$

Note that τ_m can also be interpreted as the norm of the cross product between the arm $\mathbf{r} = R(\theta)(\mathbf{p_o} - \mathbf{c})$ and the mouse force $\mathbf{F_m}$.

As in the case of translations, the scan D' is iteratively rotated until convergence of the correspondences is reached. In each iteration, the balance of torques

$$\tau_m + \tau_r = 0, \tag{22}$$

has a closed form solution. Using (20) and (21) in (22), we can obtain

$$\tan(\theta) = \frac{k_m \mathbf{r_i}^T A^T \mathbf{p_f}' + k_r \sum_{k=1}^{N} (\mathbf{d_k}')^T A^T \mathbf{m_k}'}{k_m \mathbf{r_i}^T \mathbf{p_f}' + k_r \sum_{k=1}^{N} (\mathbf{d_k}')^T \mathbf{m_k}'}. \tag{23}$$

However, this only allows us to compute θ up to a π congruence. To resolve this ambiguity, which is caused by the existence of two solutions π radians apart, one has to determine which one corresponds to a stable solution. This can be easily determined from the sign of the derivative of the total torque $\tau = \tau_m + \tau_r$,

$$\frac{\partial \tau}{\partial \theta} = -k_m \mathbf{r_i}^T R^T(\theta)\mathbf{p_f}' - k_r \sum_{k=1}^{N} (\mathbf{d_k}')^T R^T(\theta)\, \mathbf{m_k}'. \tag{24}$$

A solution is stable if and only if the sign of this derivative is negative. A positive derivative implies that a small perturbation in θ will swing the scan π radians towards the other solution. Note that (24) has opposite signs for angles θ and $\theta + \pi$, and therefore there is always a single negative solution.

The scan adjustment follows a similar algorithm as in the case of translations (Al. 2). The algorithm starts by computing the scan centroid (*ComputeScanCentroid*) and used it to center the mouse positions in the origin. In

Algorithm 2 Compute Rotation Angle.

```
function ROTATIONS($p_o, p_f, M, D'$)
    $c \leftarrow ComputeScanCentroid(D')$
    $D_{new} \leftarrow D'$
    $r_i \leftarrow p_o - c$
    $p'_f \leftarrow p_f - c$
    repeat
        $P \leftarrow ComputeClosestPoints(M, D_{new})$
        for each pair of closest points in $P$ do
            $m'_k \leftarrow m_k - c$
            $d'_k \leftarrow d_k - c$
            $P' \leftarrow (m'_k, d'_k)$
        end for
        $\theta \leftarrow ComputeRotationAngle(r_i, p'_f, P')$
        if $sign(TorqueDer(\theta, r_i, p'_f, P')) > 0$ then
            $\theta \leftarrow \theta + \pi$
        end if
        $D_{new} \leftarrow UpdateScan(D', \theta)$
    until $ErrorChange(\theta) < \xi_\theta$
    return $\theta$
end function
```

each iteration, new correspondences are obtained (*ComputeClosestPoints*) and used to compute the rotation angle (*ComputeRotationAngle*) with 23. The stable solution is then chosen (*TorqueDer*) using (24). Finally scan D' is updated (*UpdateScan*) with the new estimate of the rotation angle and the convergence is checked (*ErrorChange*). The algorithm iterates until the module of the difference between the new and the previous rotation angle falls below a threshold:

$$|\theta_i - \theta_{i-1}| < \xi_\theta, \tag{25}$$

where θ_{i-1} and θ_i are the previous and the new estimation of the rotation angle and ξ_θ is a threshold.

A homogeneous transformation reflecting the rotation performed, is then created using the value θ:

$$T_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & b_x \\ \sin(\theta) & \cos(\theta) & b_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{26}$$

$$\mathbf{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} = [I - R(\theta)]\,\mathbf{c}. \tag{27}$$

Note that translation $\mathbf{b}$ accounts for the fact that the rotation is performed with respect to center $\mathbf{c}$, rather than to the origin.

3. Results

A dataset was, initially, obtained with a Nomadic Scout robot with a Hokuyo LIDAR onboard. Each scan was associated with a robot location estimated by the robot. This estimation was performed using a simple implementation of EKF together with scan matching using the LIDAR. A partial vector map of the environment was used. As a consequence, the localization algorithm got lost upon entering an office outside of its map (upper part in Fig. 4).

In order to see how the ICP algorithm performs, we run it over the dataset of Fig. 4, obtaining the result shown in Fig. 5. The ICP was able to align most scans, however

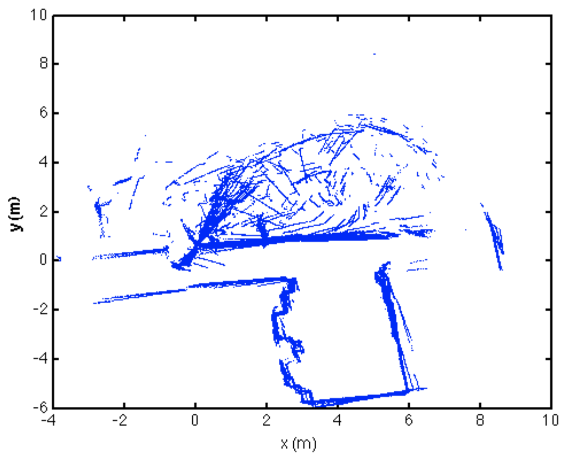

Fig. 4. Initial dataset of a map with 118 scans, obtained using a simple implementation of EKF together with scan matching using the LIDAR.

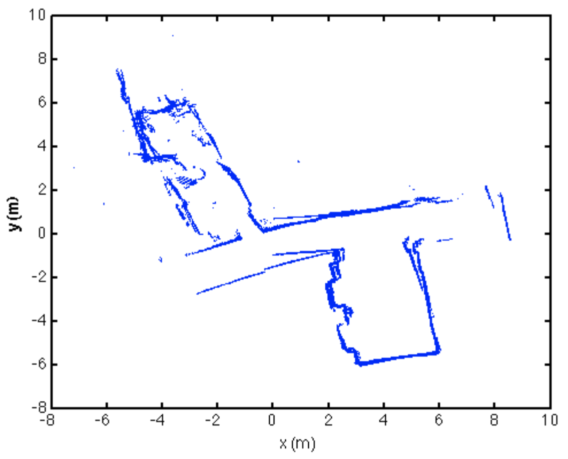

Fig. 5. Map obtained aligning the 118 frames of the initial dataset with ICP.

some of them converged to a local minimum, resulting into a wrong alignment.

In Fig. 9 and 10 is shown the alignment of two scans with and without the use of forces for the two interactive modes, translations and rotations respectively. On the left sub-figures of Fig. 9 and 10, we can see that, without the use of forces, the scan movement tends to follow the mouse cursor. On the contrary on the right side, the use of forces favours the adjustment of scan along the corridor, and disfavours the movement orthogonal to it.

To validate the proposed method a user study was conducted by taking 8 users selected among the MSc students from Instituto Superior Técnico (IST). They were asked to correct the alignment of 118 frames, from a map, in two trials: with and without the use of forces. Users had no prior knowledge of the map, and everyone performed the correction of the alignment on the same initial dataset (Fig. 4).

The user performance was measured by (i) the time they took to finish the task, and by (ii) the value of the cost function, that measures how good the corrections were made. This cost function is the sum of the cost function for all the scan pairs:

$$f = \sum_{i=1}^{n-1} J_i, \tag{28}$$

where n is the number of scans and J_i is the cost function between two scans S_i and S_{i+1}, computed from expression (3).

If the mouse force is much higher than the reaction force, users will not feel the system restraining the movement of the scan, and consequently they will have total control over the scans. On the contrary if the reaction force is much higher, the system will restrain the movement of the scans so strongly that users will not be able to move the scans as they desire. So, in order to have a good trade-off between control over the scans and help from the system, the forces proportionality constants (k_m and k_r) were empirically adjusted, so that the reaction force is slightly higher than the mouse force.

In the following experiments, the values of $k_m = 0.1$ and $k_r = 0.001$ were used, together with a closest point threshold of 0.2 meters.

After doing each one of the trials, users obtain knowledge about the initial alignment of each frame which will influence the second experiment, so we mitigate this bias by alternating the order of the experiences between users. Half of them were asked to perform the trial without using forces first, while the other half performed the other one first. Fig. 6 shows an example of a map after the interactive alignment, for a random user.

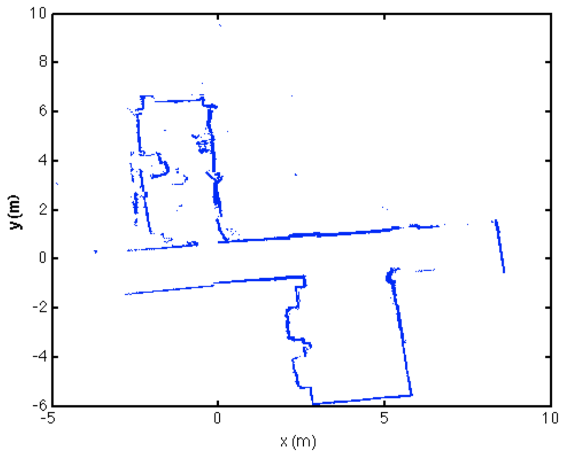

Fig. 6. One of the maps obtained by a user after using the GUI to manually align the scans (compare with Fig. 4).

The results obtained for each user are presented in Fig. 7.

Tab. 1. Average and standard deviation values for time and cost function.

	Without Forces		With Forces	
	Time (min)	Cost function Value	Time (min)	Cost function Value
μ	16.27	36.86	10.74	30.60
σ	5.45	8.09	2.77	3.33

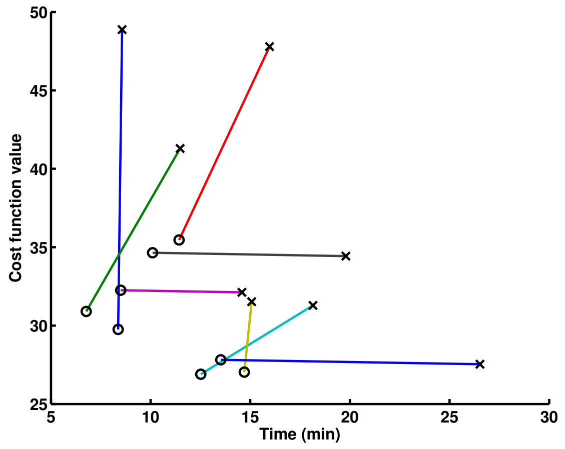

Fig. 7. Results in terms of task completion time and cost function obtained for each user. Each line corresponds to a user, and the "O" and the "X" markers are the results of using or not using forces.

As it is visible in Fig. 7 and Table 1, users were, in average, able to correct the alignment faster and better using forces than not using them. Paired t-tests over the results have shown that both the average time and average cost functions are lower when forces are used ($p < 0.005$ for time and $p < 0.03$ for cost).

At the end of the trials, users were asked to answer a questionnaire, using the well-known Likert scale as a rating criterion (from 1 to 5). The Likert statements were:

1) *The interface used for the interactive mapping was easy to use.*
2) *The use of forces makes the alignment faster.*
3) *The use of forces makes the alignment easy.*

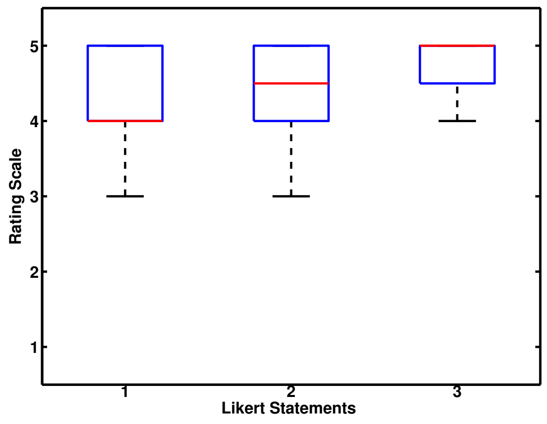

Fig. 8. Box plot of the responses to the questionnaire.

In Fig. 8 is shows the box plot responses given by the users to each statement the questionnaire. Overall, users responded positively to the usage of the GUI, in line with the goal of providing a system to facilitate manual alignment of range scans.

4. Conclusions

In this paper we have proposed interactive method: a method using virtual forces with the purpose of helping

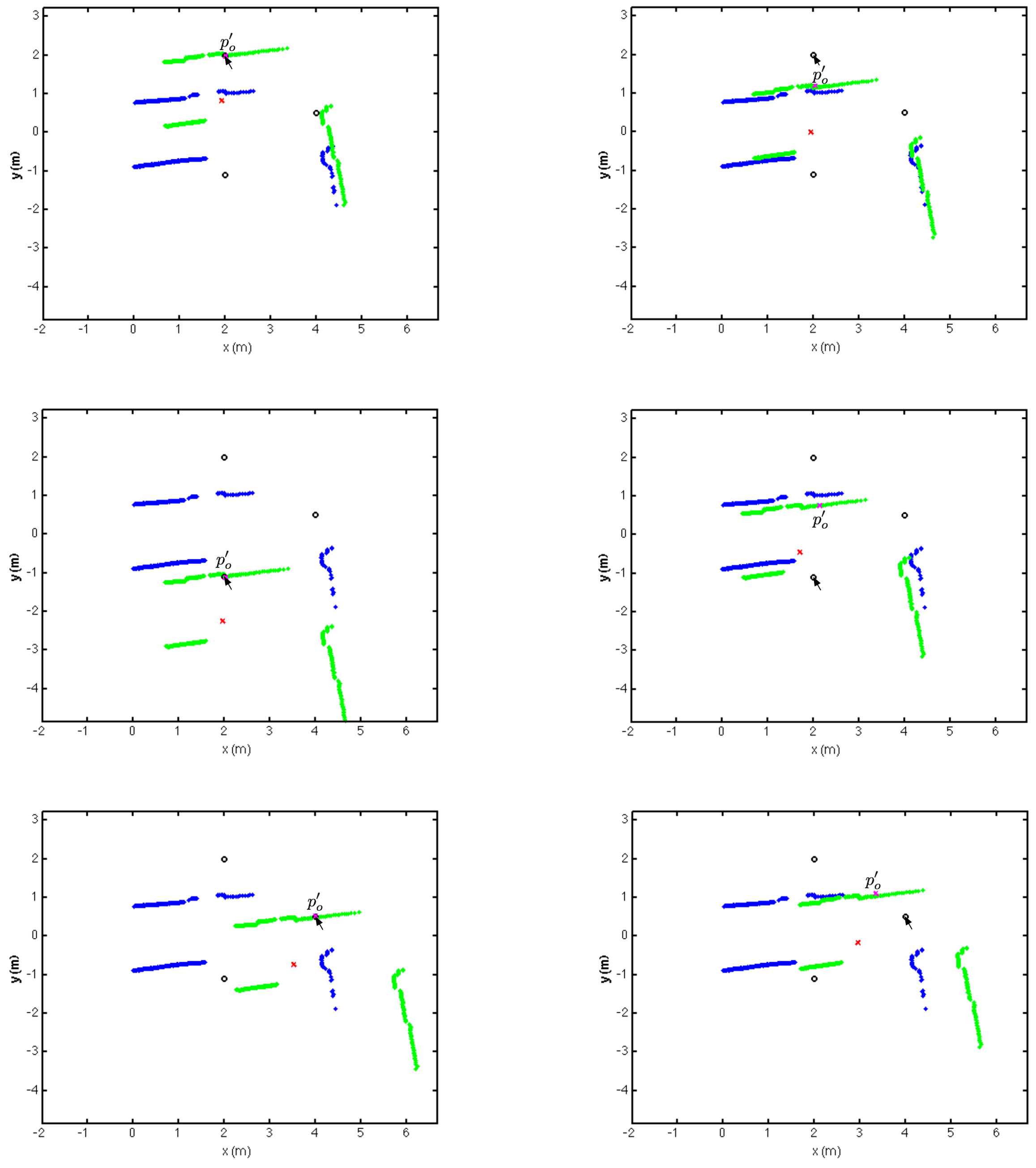

Fig. 9. Two corridors being aligned by applying translations. Left three figures: alignment without forces. Right three figures: alignment with forces. The blue/darker scan (M) is the original segment, the green/brighter scan (D') is the new one, the black arrow is the mouse current position, $\mathbf{p_o}'$ is the initial point clicked, **c** *is the green scan center of mass and the back "O" are references points.*

users to adjust the alignment of range scan data. The use of forces proved to be a valuable aid in the correction of the alignment, making it faster and more easy to use. A small user evaluation has corroborated this method. As future work we propose extending the concepts proposed in this paper to 3D.

AUTHORS
Pedro Vieira* – Institute for Systems and Robotics, Instituto Superior Técnico , e-mail: pedro.s.vieira@ist.utl.pt.
Rodrigo Ventura – Institute for Systems and Robotics, Instituto Superior Técnico, e-mail: rodrigo.ventura@isr.ist.utl.pt.

*Corresponding author

Acknowledgements

This work was supported by FCT projects [PEst-OE/EEI/LA0009/2011] and [PTDC/EIA-CCO/113257/2009].

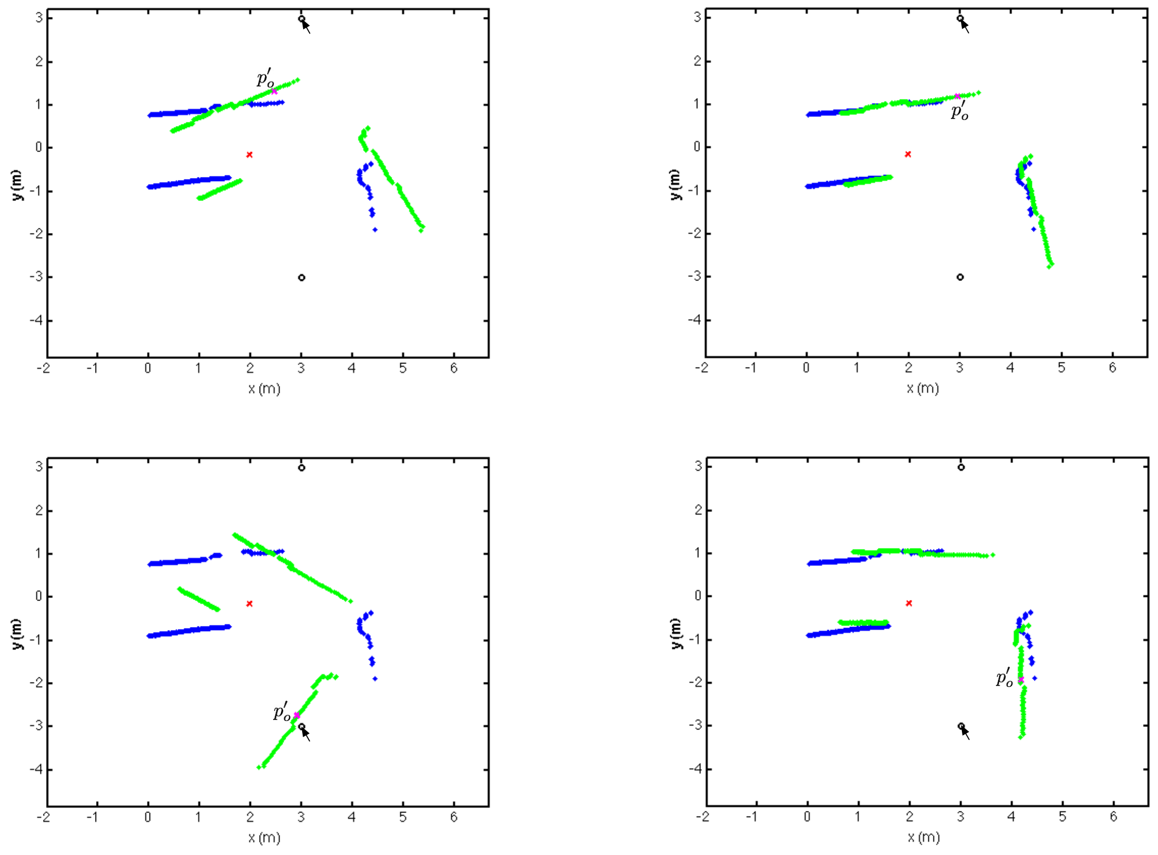

Fig. 10. Two corridors being aligned by applying rotations. Left two figures: alignment without forces. Right two figures: alignment with forces. The blue/darker scan (M) is the original segment, the green/brighter scan (D') is the new one, the black arrow is the mouse current position, $\mathbf{p_o}'$ is the initial point clicked, $\mathbf{c}$ is the green scan center of mass and the back "O" are references points.

References

[1] S. Thrun, "Robotic mapping: A survey". In: *Exploring artificial intelligence in the new millennium, G. Lakemeyer, B. Nebel (eds.)*, Morgan Kaufmann, 2002, pp. 1–35.

[2] A. Elfes, "Sonar-based real-world mapping and navigation", *IEEE J. Robot. Automat.*, vol. 3, no. 3, 1987, pp. 249–265.

[3] F. Lu, E. Milios, "Globally consistent range scan alignment for environment mapping", *Autonomous robots*, vol. 4, no. 4, 1997, pp. 333–349.

[4] D. Murray, C. Jennings, "Stereo vision based mapping and navigation for mobile robots". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'97)*, vol. 2, April 1997, pp. 1694–1699.

[5] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments". In: *The 12th International Symposium on Experimental Robotics (ISER)*, 2010.

[6] A. Elfes, "Occupancy grids: A probabilistic framework for robot perception and navigation", Ph.D. dissertation, Carnegie Mellon University, 1989.

[7] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, D. Cole, I. Reid, "Navigating, recognising and describing urban spaces with vision and lasers", *The International Journal of Robotics Research*, vol. 28, no. 11–12, 2009, pp. 1406–1433.

[8] J. Besl, N. D. McKay, "A method for registration of 3-d shapes", *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2), 1992, pp. 239–256.

[9] K. Lai, L. Bo, X. Ren, D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'11)*, 2011, pp. 1817–1824.

[10] F. Dellaert, D. Bruemmer, "Semantic slam for collaborative cognitive workspaces". In: *AAAI Fall Symposium Series 2004: Workshop on The Interaction of Cognitive Science and Robotics: From Interfaces to Intelligence*, 2004.

[11] A. Diosi, G. Taylor, L. Kleeman, "Interactive SLAM using laser and advanced sonar". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'05)*, 2005, pp. 1103–1108.