Interactive Mapping in 3D Using RGB-D Data

Pedro Vieira and Rodrigo Ventura
Institute for Systems and Robotics
Institute Superior Técnico
Av. Rovisco Pais, 1; Lisbon, Portugal
pedro.s.vieira@ist.utl.pt, rodrigo.ventura@isr.ist.utl.pt

Abstract — The task of 3D mapping indoor environments in Search and Rescue missions can be very useful on providing detailed spacial informarion to human teams. This can be accomplished using field robots, equipped with sensors capable of obtaining depth and color data, such as the one provided by the Kinect sensor. Several methods have been proposed in the literature to address the problem of automatic 3D reconstruction from depth data. Most methods rely on the minimization of the matching error among individual depth frames. However, ambiguity in sensor data often leads to erroneous matching (due to local minima), hard to cope with in a purely automatic approach. This paper is targeted to 3D reconstruction from RGB-D data, and 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 constrains human intervention along the degrees of freedom with most uncertainty. The user is able to translate and rotate individual RGB-D point clouds, with the help of a force field-like reaction to the movement of each point cloud. A dataset was obtained and used using a kinect equipped on the tracked wheel robot RAPOSA-NG, developed for Search and Rescue missions. Some preliminary results are presented, illustrating the advantages of the method.

Keywords: *3D Mapping, Interactive Alignment, Interactive Closest Points, Iterative Methods*

I. Introduction

In past few years, Search and Rescue (SaR) robots have been developed and used in missions on areas dangerous for human presence, such as inside buildings close to collapse or environments with radioactive contamination. Missions can vary from searching for victims within risky areas, to analysing certain areas for better mission planning. However, the availability of a map of these areas can improve dramatically the efficiency and effectiveness of these operations. This paper focus on the problem of constructing 3D maps of indoor areas, using a remotely operated robot. In particular, we target maps that include both depth and color data. In order to make maps for posterior analysis and mission planning, robots must be equipped with capable sensors. Methods to address the problem of scan-matching have been proposed and used in many applications, in particular for 3D mapping of an environment (see [1] for a review). These methods vary both in terms of sensors used (e.g., sonars [2], LIDAR [4], vision [5], and more recently the Kinect [6]), and in methodologies (e.g., probabilistic [3], scan matching [4]). However, most of these methods are prone to local minima, originated, for instance, from ambiguity or from locally periodic patterns in the environment.

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

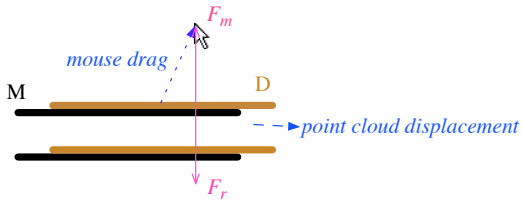

Fig. 1. Top view of 2 corridors. The scan D is adjusted such that the reaction force F_r, computed from the cost function gradient, balances the force F_m imposed by the mouse drag.

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 invited to interact with the matching process, by adjusting the match of individual 3D pairs of scans. This adjustment is however constrained by favoring adjustments along the directions of greater ambiguity. Take for instance a case of pairs of identical scans taken from an homogeneous corridor (schematized in Fig. 1, a pair of 2D scans are used here for simplicity sake). The system should favor the adjustment of the scans along the corridor, while disfavoring 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 scan D, using the mouse, a corresponding virtual force F_m is produced (proportional to the drag vector). Opposing it, a reaction force 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 can be seen as a reaction force 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, $F_m + F_r = 0$. In the corridor example of Fig. 1, the reaction force is (mostly) orthogonal to the corridor axis.

The Interactive Closest Points (ICP) algorithm (see [7]) is used to obtain the initial rigid transformation between M and D automatically. Then the proposed method is used to correct any ambiguity in the alignment. Note that there are other automatic scan-matching algorithms[1] and other variants of ICP [8] that have better performance than the original ICP. We use the original ICP together with efficient correspondences computation, for the sake of simplicity.

This paper is organized as follows. In Section II the method for 3D interactive mapping is proposed, followed by Section III showing the preliminary results based on the user experience with the alignment. Finally, Section IV draws some conclusions and discuss future work directions.

II. 3D Interactive Alignment

Consider two point clouds in 3D space, $M = \{m_k\}$ and $D = \{d_k\}$, for $m_k, d_k \in \mathbb{R}^3$, and an initial rigid transformation (R,t) which align D with M, where R is a rotation matrix and t is a translation vector. This transformation can be initialized to a default value (e.g. $R = I_{3\times3}$ and $t = (0,0,0)^\top$) or computed with an alignment algorithm, like ICP algorithm. By applying this transformation to D, a transformed point cloud, D' is obtained. The two point clouds (M and D') are then presented in a viewer (GUI) for alignment analysis. Then, the user interacts with the viewer using common computer mouse and keyboard, and apply either translations or rotations to D'. These actions are carried out separately using a designated key to choose which mode to use.

Point clouds are aligned by balancing a virtual force caused by a mouse drag (mouse force) with a reaction force produced by a potential field. In 3D, we only have access to the coordinates of the pixel of the mouse current position in the image plane, therefore the mouse force is defined on a 3D plane parallel to the image plane. The drag is defined by two 3D points, p_o and p_f, corresponding to the initial and final drag points. When the user clicks on pointcould D', the initial point p_o is set to the point on D' which projection on the image plane is closest to the mouse point clicked by the user. As the user drags the mouse, p_f is defined by the projection of the new mouse point onto the plane parallel to the image plane that crosses p_o (Fig. 2). As point cloud D' moves during interaction, the point cloud point corresponding to p_o also moves. Consider that point cloud D' movement is defined by a rigid transformation, defined by a rotation matrix R, a rotation center c, and a translation t. Then, point p_o is transformed into $p'_o = R(p_o - c) + c + t$. Now, we define a potential function J_m that grows with the distance between points p'_o and p_f:

$$J_m = \frac{1}{2}\|p_f - p'_o\|^2, \tag{1}$$

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

[1]see RGBDSLAM in http://www.ros.org/wiki/rgbdslam

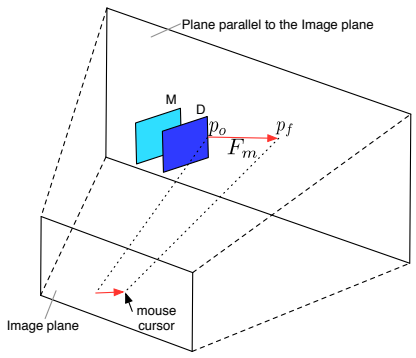

Fig. 2. 3D representation of the mouse force, defined by p_o and p_f. Force is applied to point cloud D.

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

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

where $\{m_k\}$ and $\{d_k\}$ are pairs of closest points from M and D' respectively, and N is the number of these pairs. This function is identical to the cost function used in ICP. The closest points are computed in the same fashion as in ICP, and only the pairs sufficiently close are considered.

Note that the correspondences are computed every time the forces are computed. These correspondences may not match the true ones. However, they are essential to produce attraction forces among point clouds.

The interaction between the user and the point cloud, through a computer mouse, is restricted to a plane parallel to the image plane, due the fact it only has two degrees of freedom, therefore translations are performed in this plane (see II-A). Performing rotations in 3D with a computer mouse is not a trivial task, so, in our interactive alignment, the user is allowed to choose among tree types of rotations, based on previous studies concerning human-computer interaction with 3D objects:

1) Rotation without restrictions: in this mode the three degrees of freedom are taken into account when computing the rotations matrix (see II-B);
2) Rotation restrict to a plane: in this mode rotations are performed around an axis orthogonal to the plane parallel to the image plane (see II-C);
3) Rotation using a virtual sphere: in this mode the point clicked is projected onto the image plane and used alongside the mouse current position in the image plane to compute the rotation axis and the angle each time the mouse moves (see II-D).

We expect to conduct in the future user studies to evaluate the performance of each one of these modes.

A. Translations

In this mode, the alignment consists in a translation by t. Thus R is the identity, and the potential function (1) is simplified. The mouse force F_m is computed from the gradient of the cost function (1) with respect to the translation t:

$$F_m = -k_m \nabla_t \, J_m|_{R=I} = k_m \, (p_f - p_o - t) \,, \tag{3}$$

where k_m is the proportionality constant.

Opposing this force, a reaction force F_r is computed from the gradient of the function (2) with respect to the translation t:

$$F_r = -k_r \nabla_t \, J_r|_{R=I} = k_r \sum_{k=1}^{N} (m_k - d_k - t) \,, \tag{4}$$

where k_r is the proportionality constant,

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

$$F_m + F_r = 0, \tag{5}$$

with respect to t. This equation has the following algebraic closed form solution:

$$t = \frac{k_m(p_f - p_o) + k_r \sum_{k=1}^{N}(m_k - d_k)}{k_m + N k_r}. \tag{6}$$

The interactive scan adjustment proceeds according to the following algorithm:

1) Compute translation t according to (6);
2) Compute the new correspondences $\{m_k\}$ and $\{d_k\}$ from scans M and D';
3) Unless the correspondences are the same, go to step 1).

The obtained $t = [t_x \; t_y \; t_z]^T$ corresponds to the homogeneous transformation:

$$T_t = \left[\begin{array}{c|c} I_{3\times 3} & t \\ \hline 0 & 1 \end{array} \right], \tag{7}$$

where $I_{3\times 3}$ is a 3×3 identity matrix.

B. Rotations

Rotation mode inflict a rotation of point cloud D', with respect to the center of mass c. The center of mass of a point cloud is set to its centroid. When the user clicks on the point cloud D' and attempts to drag it, a force F_m is created using (1), for $t = 0$ and $R = R(\alpha, \beta, \gamma)$. However, unlike translations, the balance is formulated here in terms of virtual torques.

The mouse torque τ_m is the gradient of the cost function (1) computed around rotation R. We consider infinitesimal rotations α, β and γ, along axis x, y, and z.

$$\tau_m = -k_m \nabla_{\alpha,\beta,\gamma} \, J_m|_{t=0,\alpha=\beta=\gamma=0} \,. \tag{8}$$

The opposing torque τ_r is the gradient of the cost function (2) with respect to α, β and γ:

$$\tau_r = -k_r \nabla_{\alpha,\beta,\gamma} \, J_r|_{t=0,\alpha=\beta=\gamma=0} \,. \tag{9}$$

Both gradients can be computed using the chain rule:

$$\nabla_{\alpha,\beta,\gamma} \, J_m|_{t=0,\alpha=\beta=\gamma=0} = \left[tr \left[(d_m)^T \, d_R \right] \right], \tag{10}$$

$$\nabla_{\alpha,\beta,\gamma} \, J_r|_{t=0,\alpha=\beta=\gamma=0} = \left[tr \left[(d_r)^T \, d_R \right] \right], \tag{11}$$

where

$$d_m = \frac{\partial J_m}{\partial R(\alpha,\beta,\gamma)}, \; d_r = \frac{\partial J_r}{\partial R(\alpha,\beta,\gamma)}, \; d_R = \frac{\partial R(\alpha,\beta,\gamma)}{\partial x} \tag{12}$$

with $x \in \{\alpha, \beta, \gamma\}$.

Derivatives (10) and (11) are performed with respect to each one of the rotation axes, around a nominal rotation R. This can be formulated in Lie algebraic terms, as the tangent space of Lie group SO(3) at point R. This tangent space is spawned by matrices A_α, A_β, and A_γ, defined by

$$A_\alpha = \left[\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{smallmatrix} \right], \quad A_\beta = \left[\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{smallmatrix} \right], \quad A_\gamma = \left[\begin{smallmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix} \right],$$

while the derivative of the rotation matrix is

$$\frac{\partial R(\alpha,\beta,\gamma)}{\partial x} = A_x \, R, \tag{13}$$

for $x \in \{\alpha, \beta, \gamma\}$.

The partial derivatives d_m and d_r are trivially obtained by computing the derivative of (1) and (2) respectively, with respect to R:

$$d_m = -p'_f r_i^T, \tag{14}$$

$$d_r = -\sum_{k=1}^{N} m'_k (d'_k)^T, \tag{15}$$

where $r_i = (p_o - c)$, $p'_f = (p_f - c)$, $d'_k = (d_k - c)$ and $m'_k = (m_k - c)$.

As in the case of translations, the point cloud D' is iteratively rotated until convergence of the correspondences is reached. In each iteration, due the non-linearity of the system, a suitable solver is used to balance the two torques:

$$\tau_m + \tau_r = 0. \tag{16}$$

Using the results from (13), (14) and (15), the balance of torques consists in solving the following system of equations with respect to R:

$$\mathrm{tr} \left[\left(k_m r_i \left(p'_f \right)^T + k_r \sum_{k=1}^{N} d'_k \left(m'_k \right)^T \right) A_x R \right] = 0, \tag{17}$$

$$R^T R = I \,, \quad x = \alpha, \beta, \gamma. \tag{18}$$

The scan adjustment follows a similar algorithm as in the case of translations:

1) Compute rotation R by numerically solving[2] equations (17) and (18) simultaneously.
2) Compute the new correspondences $\{m_k\}$ and $\{d_k\}$ from scans M and D';
3) Unless the correspondences are the same, go to step 1).

[2]A variation of the Levenberg Marquardt algorithm was used, as implemented in Eigen library (http://eigen.tuxfamily.org).

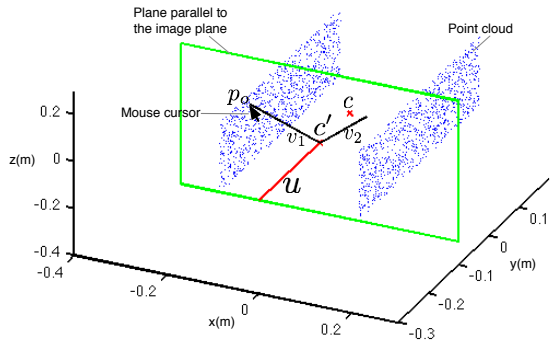

Fig. 3. 3D representation of the axis of rotation, u, orthogonal to a plane parallel to the image plane.

Note that step 2 and 3 are exactly the same as in translations.

A rotation transformation is then created using the solution matrix R,

$$T_R = \left[\begin{array}{c|c} R & b \\ \hline 0 & 1 \end{array}\right], \tag{19}$$

$$b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [I - R]\ c. \tag{20}$$

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

C. Restricted rotation mode

An alternative mode for rotating point clouds is to make the rotations around an axis perpendicular to the plane parallel to the image plane, and centered in the center of mass projected on this plane (Fig. 3). The Degrees of Freedom (DoFs) are reduce from 3 to 1. The Rotation matrix is given by the Rodrigues' rotation formula according to which, given a unit vector $u = (u_x, u_y, u_z)$, where $u_x^2 + u_y^2 + u_z^2 = 1$, the matrix for a rotation by an angle θ about an axis in the direction of u is:

$$R = I\cos\theta + \sin\theta\, U_1 + (1 - \cos\theta)\, U_2, \tag{21}$$

where

$$U_1 = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \text{ and } U_2 = \begin{bmatrix} u_x^2 & u_x u_y & u_x u_z \\ u_x u_y & u_y^2 & u_y u_z \\ u_x u_z & u_y u_z & u_z^2 \end{bmatrix}.$$

When the user clicks on the point cloud D' and attempts to drag it, a force F_m is created using (1), for $t = 0$, $R = R(\theta)$ and $c = c'$, where c' is the centroid project onto the plane. The axis orthogonal to the plane, u, can be obtained by computing the cross product of two vectors belonging to the plane:

$$u = v_1 \times v_2, \tag{22}$$

where v_1 is the vector defined by the user clicked point, p_o, and the centroid project onto the plane, c' and v_2 is a vector orthogonal to v_1 and defined by a point belonging to the plane and c'. After being computed, vector u, needs to be normalized to meet the constrain $u_x^2 + u_y^2 + u_z^2 = 1$:

$$u = \frac{u}{\|u\|}. \tag{23}$$

The mouse torque τ_m is the gradient of the cost function (1) with respect to θ:

$$\tau_m = -k_m\, \nabla_\theta\ J_m|_{t=0}\,. \tag{24}$$

The opposing torque τ_r is the gradient of the cost function (2) with respect to θ:

$$\tau_r = -k_r\, \nabla_\theta\ J_r|_{t=0}\,. \tag{25}$$

Both gradients can be computed using the chain rule:

$$\nabla_\theta\ J_m|_{t=0} \quad = \quad tr\left[(d_m)^T\, d_R\right], \tag{26}$$

$$\nabla_\theta\ J_r|_{t=0} \quad = \quad tr\left[(d_r)^T\, d_R\right]. \tag{27}$$

where in this case:

$$d_m = \frac{\partial J_m}{\partial R\,(\theta)},\ d_r = \frac{\partial J_r}{\partial R\,(\theta)},\ d_R = \frac{\partial R\,(\theta)}{\partial \theta}. \tag{28}$$

The partial derivative d_R can be computed by making the derivative of (21) with respect to θ:

$$d_R = \frac{\partial R\,(\theta)}{\partial \theta} = U_1\ \cos\,(\theta) - (I - U_2)\ \sin\,(\theta)\,. \tag{29}$$

The partial derivatives d_m and d_r have the same result as (14) and (15) respectively, the only difference is that $r_i = (p_o - c')$, $p'_f = (p_f - c')$, $d'_k = (d_k - c')$ and $m'_k = (m_k - c')$.

The balance of torques (16) has a closed form solution that can be obtained using the results from (14), (15) and (29)

$$\tan(\theta) = \frac{k_m r_i^T U_1^T p'_f + k_r \sum_{k=1}^{N} (d'_k)^T U_1^T m'_k}{k_m r_i^T U_2' p'_f + k_r \sum_{k=1}^{N} (d'_k)^T U_2' m'_k}, \tag{30}$$

where $U_2' = (I - U_2)^T$. However, this only allows us to compute θ up to a π congruence. To disambiguate among both solutions, 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 $\tau = \tau_m + \tau_r$,

$$\frac{\partial \tau}{\partial \theta} = -H_1 \cos\theta - H_2 \sin\theta, \tag{31}$$

where

$$H_1 \quad = \quad k_m r_i^T U_2' p'_f + k_r \sum_{k=1}^{N} (d'_k)^T U_2' m'_k, \tag{32}$$

$$H_2 \quad = \quad k_m r_i^T U_1^T p'_f + k_r \sum_{k=1}^{N} (d'_k)^T U_1^T m'_k. \tag{33}$$

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 point cloud π radians towards the other solution. Note that (31) has opposite signs for angles θ and $\theta + \pi$, and therefore there is always a single negative solution.

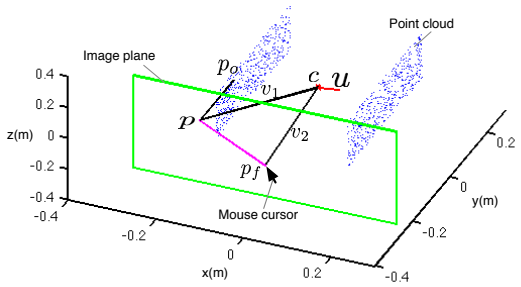

Fig. 4. 3D represention of the axis rotation computed with the point p_o projected onto the image plane and the current mouse position in the image plane.

The scan adjustment follows a similar algorithm as in the case of translations:

1) Compute c' by projecting the point cloud centroid onto the plane parallel to the image plane;
2) Compute the rotation axis, u, using (22) and (23);
3) Compute rotation θ according to (30), choosing the solution θ or $\theta + \pi$ with negative derivative (31);
4) Compute the new correspondences $\{m_k\}$ and $\{d_k\}$ from scans M and D';
5) Unless the correspondences are the same, go to step 3).

A rotation transformation is then created using the value θ, in the Rodrigues equation (21) and used to create the homogeneous transformation, T_θ:

$$T_\theta = \left[\begin{array}{c|c} R(\theta) & b \\ \hline 0 & 1 \end{array}\right], \tag{34}$$

$$b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [I - R(\theta)]\, c'. \tag{35}$$

D. Rotation using a virtual sphere

In this rotation mode, rotations are also performed around a rotation axis. The rotation matrix is also computed using the Rodrigues equation (21), however the axis of rotation, u, is computed using different vectors. When the user clicks on a point of the point cloud, this point is projected onto the image plane. Each time the mouse moves the rotation axis is computed by doing the cross product between vectors v_1 and v_2 (Fig.4):

$$u = v_1 \times v_2, \tag{36}$$

with $v_1 = p - c$ and $v_2 = p_f - c$, where p is the point p_o projected onto the image plane, p_f is the mouse current position in the image plane and c is the point cloud center of mass. The rotation axis is then normalized using (23). The point cloud movement will have a similar behavior to a virtual sphere (see [9] for more information).

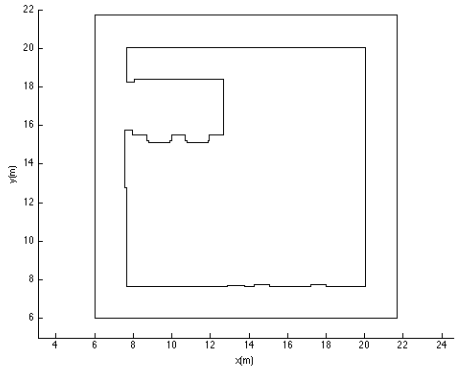

Fig. 5. 2D representation of the map.

The mouse torque τ_m, the opposing torque τ_r, and the balance of the torques $\tau_m + \tau_r = 0$, are computed in the same way as in II-C, the only difference is that, in this case, $r_i = p - c$ and the rotation axis (36) are used in the calculations.

The scan adjustment follows a similar algorithm as in II-C.

1) Project point p_o onto the image plane;
2) Move mouse cursor in the image plane to obtain point p_f;
3) Compute the rotation axis, u, using (36) and (23);
4) Compute rotation θ according to (30), choosing the solution θ or $\theta + \pi$ with negative derivative (31);
5) Compute the new correspondences $\{m_k\}$ and $\{d_k\}$ from scans M and D';
6) Unless the correspondences are the same, go to step 4).

A rotation transformation is then created using the value θ, in the Rodrigues equation (21) and used to create the homogeneous transformation with (34) and (35).

III. Preliminary Results

A dataset was obtained using RAPOSA-NG[3] equipped with a Microsoft Kinect sensor, which was used to get a sequence of RGB-D point clouds. Figure 5 shows the 2D layout of the floor where the dataset was collected. Figure 6 illustrates the results: (1) shows this collection after an initial pass through ICP. The resulting alignment is visibly erroneous, due to local minima. A user, with average level of computer knowledge, was then asked to employ the proposed method to interactively align the point clouds: Figure 7-(1) and (2) show an arbitrary pair of (consecutive) point clouds, before and after the alignment. This alignment demanded for a translation and a rotation: Figure 6-(2) shows the final result after the user aligned, in a pair by pair basis, all pairs of frames. The obtained result has the same shape as the 2D map in Figure 5, although, kinect has enormous error associated to the depth data, making the map blurred.

In qualitative terms, the user has found the proposed interactive method useful. Although the user had no prior knowledge

[3]http://raposa.isr.ist.utl.pt

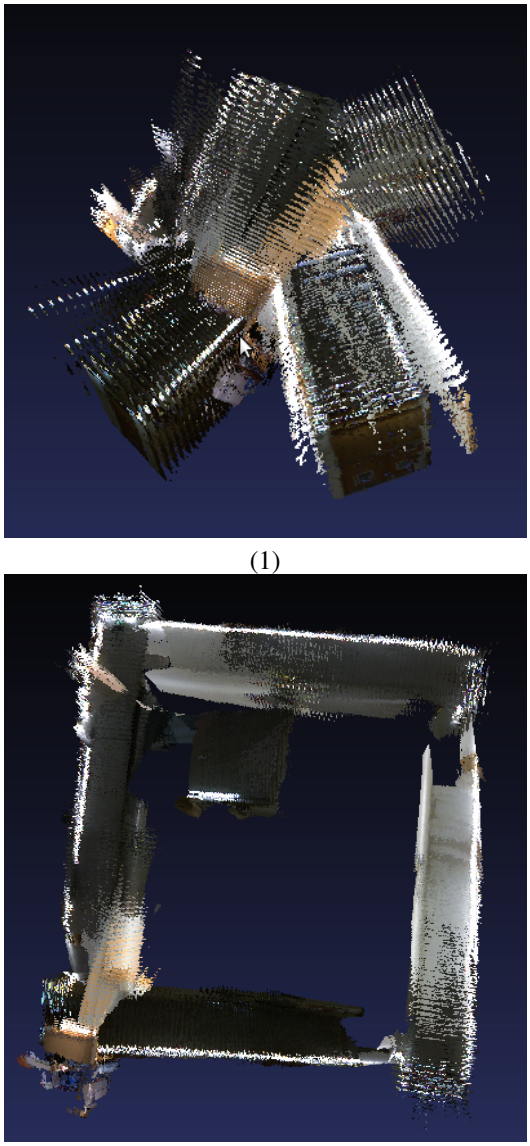

Fig. 6. From top to bottom: (1) initial 3D map after applying ICP to the raw RGB-D data; and (2) final map after interactive alignment

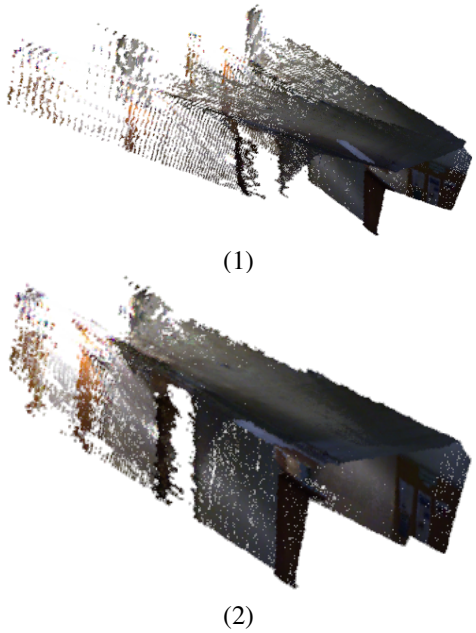

Fig. 7. From top to bottom: (1) a pair of point clouds from the initial map; and (2) the same pair after interactive alignment.

of the correct alignment, he was able to correct the alignment of the scans by identifying certain patterns in each pair of scans that match together. He showed preference in making rotations restricted to a plane and using a virtual sphere, because these are easier to control and more intuitive. However, when using forces, the other rotation mode performs better when the two point clouds are nearly aligned, due the fact of having three DoFs.

IV. Conclusion

In this paper we have proposed a method using virtual forces with the purpose of aiding users to manually adjust the alignment of RGB-D point clouds. The 3D interactive alignment with the use of forces proved to be a valuable aid in the correction of the alignment. Rotations without restrictions behaved better then the other rotation modes when points clouds were near to each other, but it was the most difficult mode to control. Rotation restricted to a plane was the easiest mode to control with the mouse, due the two degrees of freedom, but takes more time to perform the alignment. Rotations using a virtual sphere were found intuitive for the users and easy to control, but using the interactive alignment they behaved better when making smalls adjustments. As future work we propose making a more extensive user study of the proposed method, namely to evaluate the rotation methods. We also intend to use a localization method for the robot to get better initial estimates of the point cloud poses.

References

[1] S. Thrun, "Exploring artificial intelligence in the new millennium", Morgan Kaufmann, ch. Robotic mapping: A survey, pp. 1–35, 2002.

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

[3] A. Elfes, "Occupancy grids: A probabilistic framework for robot perception and navigation". Ph.D. dissertation, Carnegie Mellon Uni- versity, 1989.

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

[5] D. Murray and 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, pp. 1694–1699, April 1997.

[6] 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.

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

[8] A. Segal, D. Haehnel, and S. Thrun, Generalized-icp, in Proceedings of Robotics: Science and Systems, Seattle, USA, June 2009.

[9] R. Bade, F. Ritter and B. Preim, "Usability Comparison of Mouse-Based Interaction Techniques for Predictable 3d Rotation", In Proceedings of the 5th international conference on Smart Graphics, pp. 138–150, 2005.