2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Visual Coordination Task for Human-Robot Collaboration

Maram Khatib Khaled Al Khudir Alessandro De Luca

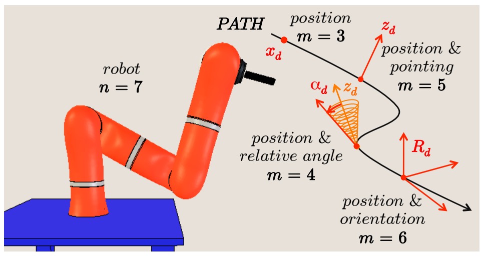
***Abstract*— In the framework of Human-Robot Collaboration, a robot and a human operator may need to move in close coordination within the same workspace. A contactless coordinated motion can be achieved using vision, mounting a camera either on the robot end-effector or on the human. We consider here one instance of such a visual coordination task, with the robot end-effector that should maintain a prescribed position with respect to a moving RGB-D camera while pointing at it. For the 3D localization of the moving camera, we compare three different techniques and introduce some improvements to the best solution found for our application. For the motion tracking problem, we introduce a relaxed version of the pointing part of the task. This allows to take advantage of the redundancy of the robot, distributing the control effort over the available degrees of freedom. The effectiveness of the proposed approach is shown by V-REP simulations and experiments with the 7-dof KUKA LWR manipulator.**

I. Introduction

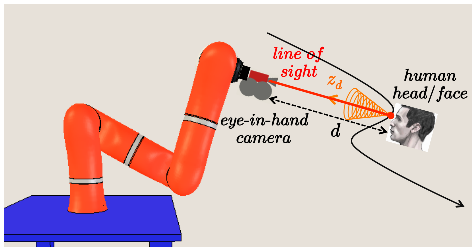
Safe handling of human-robot interaction tasks can be achieved within a hierarchical control architecture organized in three layers [1]. Each layer addresses some desired robot behavior in a way that is consistent through the layers. From the bottom to the top, the three layers are concerned with: *safety*, e.g., having the robot react to undesired contacts, which are detected without using extra sensors [2]; *coexistence*, e.g., sharing the same workspace while continuously avoiding collisions [3]; *collaboration*, e.g., engaging an intentional contact with controlled exchange of forces between robot and (human) environment [4]. Beside such a physical collaboration, contactless collaboration could also be established, when the task has to be realized in a coordinated way by the robot and the human.

Spatial and temporal motion coordination can be obtained via direct and explicit communication, such as using gestures and/or voice commands [5], or by indirect communication, such as recognizing intentions [6], raising the attention with legible action [7], or passively following the human motion. Each type of collaboration raises different challenges. Indeed, as a common feature, the robot should always be aware of the existence of the human in the workspace, and specifically of the pose of his/her body parts. Use of laser range measurements [8] and of vision/depth cameras are the preferred choices for this purpose, followed by the extraction of the human pose from the sensor data [9]. Based on this information, a coordinated motion task can be defined online by requiring the robot to track some human feature in a specific way.

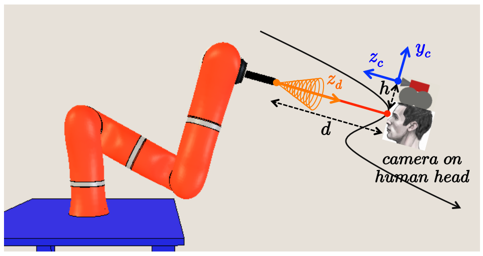
The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Universita di Roma, Via Ariosto 25, 00185 Roma, Italy ({khatib,alkhudir,deluca}@diag.uniroma1.it). This work is supported by the EC, within the H2020 project 637080 SYMPLEXITY.



(a)



(b)



(c)

Fig. 1: (a) Different types of Cartesian motion tasks, with their dimension m. (b) Visual coordination task with the camera on the robot, pointing at the moving human head/face. (c) Camera on the moving human head, with the robot end-effector pointing at it. The cones represent the relaxation of the pointing task by some relative angle α_d.

In this paper, we consider a contactless collaborative scenario with indirect communication and address the motion control problem for a visual coordination task in which a robot should track the motion of the human head (for instance, to show an item held by its end effector). In Sec. II, we formulate and detail alternatives for the visual coordination task. In particular, a RGB-D camera will be carried by the human (say, mounted on a helmet) to establish visual coordination. Section III describes the methods tested for solving efficiently the 3D-localization of the moving camera. The control problem is discussed in Sec. IV, proposing two kinematic control laws that take advantage of the available robot redundancy. Results of V-REP simulations and of an experiment with a KUKA LWR arm are reported in Sec. V.

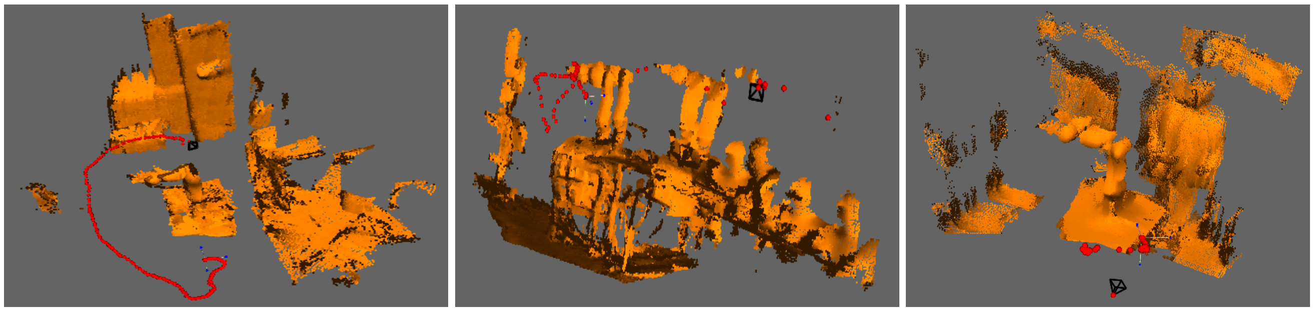

(a) (b) (c)

Fig. 2: Outputs with the NICP method in three different operative conditions: (a) slow camera moving in static environment; (b) faster motion; (c) static camera in dynamic environment. The red dots represent the camera pose estimation and the yellow parts are the 3D reconstruction of the camera scenes.

II. Visual Coordination Task

For a robot with n joints, we can define a m-dimensional task to be executed. If $n > m$, the robot will be kinematically redundant for the given task. Figure 1(a) illustrates preliminarily some typical motion tasks, with their dimension m: following the position of a reference point $\boldsymbol{x}_d$ along a path ($m = 3$), adding to this also a desired pointing direction defined by a unit vector $\boldsymbol{z}_d$ ($m = 5$ in total), or specifying in addition a complete desired orientation through a moving reference frame ($m = 6$). Further, we introduce a situation that relaxes the pointing task, while still keeping some control on it: the actual pointing direction of the robot end effector (defined by a unit vector $\boldsymbol{z}_e$) is allowed to stay within a cone with apex at $\boldsymbol{x}_d$, axis $\boldsymbol{z}_d$, and apex angle $\alpha > 0$. This is indeed an inequality constraint which does not increase the task dimension, as it happens in the case of an additional equality constraint. However, inequalities are more cumbersome to be handled within the kinematic task formalism [10]. Therefore, we consider a slightly stronger definition by imposing that the relative angle between $\boldsymbol{z}_e$ and $\boldsymbol{z}_d$ takes a (small) constant value α_d: pointing in a direction that belongs to the cone surface is then a one-dimensional task. Combining this with the positional task gives $m = 4$.

With the above in mind, our visual coordination task between the human (head/face) and the robot can be formulated in two almost symmetric ways. In Fig. 1(b), a camera is placed on the end effector (*eye-in-hand*) of the robot, whose motion needs to be controlled so as to search for the human face, point at it (with the above defined relative angle relaxation), and follow the human motion at a desired distance d along the camera line of sight. In this case, the pose of the camera is known from the robot direct kinematics. Alternatively, a camera can be mounted on the human head (say, on a helmet at height h above the eyes) as in Fig. 1(c). In this case, the moving (*eye-to-hand*) camera needs to be continuously localized with respect to the world frame. The robot will receive this information and move accordingly in order to achieve coordination. This is the situation considered in the rest of the paper, with the visual coordination task being of dimension $m = 4$, while the robot joint space is of dimension $n = 7$ for a KUKA LWR robot.

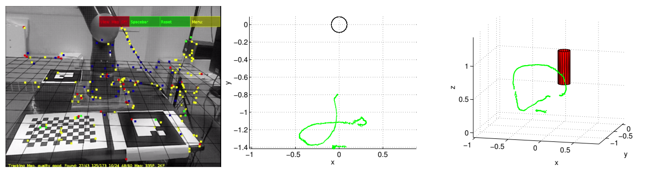
III. Camera Localization

Camera localization may be performed with vision-based or LiDAR-based methods. Vision-based methods either locate fiduciary markers using feature recognition algorithms from computer vision, or work in markerless fashion by estimating the camera pose by features extraction and point correspondences in stereo images [11]. On the other hand, LiDAR-based methods localize a RGB-D sensor based on the Iterative Closest Point (ICP) algorithm [12], which compares the current Point Cloud with a reference cloud (typically, the initial one). For our dynamic camera localization problem, we have tested and compared three different methods to find the best one that fits this application.

A. NICP

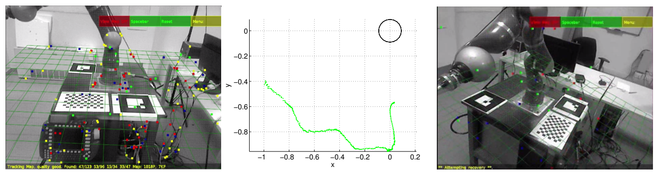
The first (markerless) method considered was the Normal Iterative Closest Point (NICP) [13], which solves the Point Cloud Registration (PCR) problem, i.e., it finds the transformation that aligns at best the common parts of two point clouds. PCR is used in 2D- or 3D-surface reconstruction, in robot localization, path planning and many other applications. NICP considers each point together with some local features of the surface, and takes advantage of the 3D structure around the points for guiding the data association between the two point clouds. Moreover, it is based on a least-squares formulation of the alignment problem, which minimizes an augmented error metric depending on point coordinates as well as on surface characteristics.

We have tested the NICP method for tracking the motion of a depth camera in different operative conditions. When the camera moves slowly (at about 0.14 m/s) in a static environment, Figure 2(a) shows that clearly NICP is able to track well the camera motion. When the camera moves slightly faster than before (at about 0.2 m/s, i.e., 40% faster) but again in a static environment, there is a duplication of some parts in the 3D map, as shown in Fig. 2(b), which implies an inaccurate camera pose estimation. In the last test, the camera was held at rest in a dynamic environment. Figure 2(c) shows that different pose estimations that do not reflect the static condition of the real camera. As a result, NICP is not suitable for our goal unless the user (carrying



(a) (b) (c)

Fig. 3: First PTAM test: (a) represents the tracking map and the features; in the top view (b) and 3D-view (c), the green dots represent the estimated path of the camera and the circle/cylinder denotes the robot position.



(a) (b) (c)

Fig. 4: Second PTAM test: (a) initialization view; (b) path of the camera; (c) camera view when the method fails.

the camera) moves very slowly in an environment which is otherwise static.

B. PTAM

The second method tested was Parallel Tracking and Mapping (PTAM) [14]. Although PTAM was originally designed for Augmented Reality (AR), its parallel framework enables fast camera localization in a small, but otherwise unknown environment. This vision-based method does not require markers, pre-made maps, or known templates.

We have used PTAM to estimate the pose of a RGB camera moving around the desired robot workspace. An initialization phase, in which the camera must be translated between the first two key-frames, is mandatory before the tracking phase can start. In a first test (Fig. 3), the camera keeps moving in the same field of view seen in the initialization phase and the method is able to track the camera motion satisfactorily. In the second test (Fig. 4), starting from the same previous initialization view, the camera was moved around the whole workspace. The method fails to estimate the camera pose as soon as its field of view exits from the one covered in the initialization phase (thus, the method strongly depends on this phase). As a result, PTAM is a good method for our tracking purposes only when the environment is relatively small and quasi-static.

C. ARToolKit

The last camera tracking test was done with the ARToolKit library [15], which is mainly used for developing AR applications. The algorithms in this computer vision library produce in fact a good solution to the problem of calculating in real time the user's viewpoint. Adding a simple calibration setup, is can be used aslo to determine the pose of a RGB camera relative to a physical marker, which can be chosen as a 2D black square card with a special pattern.

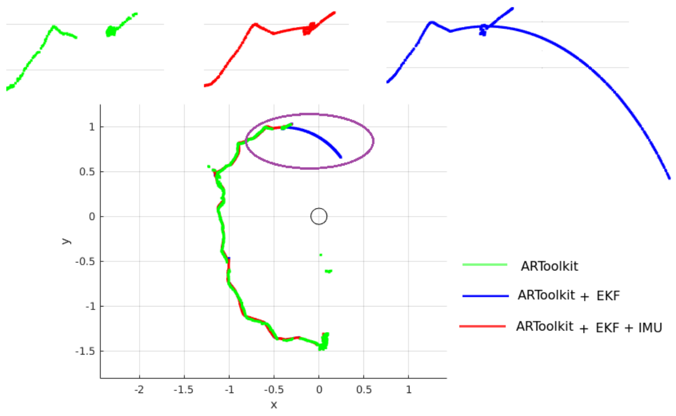


Fig. 5: The estimated trajectory of a moving camera obtained using the ARToolkit method alone (green line), and with two additional enhancement techniques (blue line = with EKF; red line = with EKF and IMU). The three top figures are expanded views of the paths inside the violet ellipse. For clarity, just a 2D projection on a horizontal plane is shown, instead of the full 3D trajectories.

In our tests, three different markers have been added to the workspace, placed on the supporting table of the robot manipulator (see also Fig. 11). Each marker has its own features (location, size, and pattern). Using the associated homogeneous transformation matrices, it is then easy to obtain online the pose of a moving camera with respect to the world frame. During the multiple experiments done, the ARToolKit performance was extremely good, as long as at least one marker was found in the camera field of view. In Fig. 5, the green line represents the path of the camera estimated with the ARToolKit method in one of the experiments. When there is no marker in the view or when this is not sufficiently clear, discontinuous tracts appear along the estimated motion path together with a number of outlier points. To address this problem, an Extended Kalman Filter (EKF) can be used in the processing of visual data, and a (cheap) Inertial Measurement Unit (IMU) can be added to the camera hardware. When the camera is in motion, the EKF eliminates discontinuities and outliers. However, when the camera stops and no marker is in the field of view, the EKF will return false camera pose estimations, i.e., the extra blue line in Fig. 5. This behavior is discarded when resorting to the IMU estimation, see the red line in Fig. 5.

From the obtained results and discussion, to address the localization problem of the moving camera in our application, the best solution would be to combine the ARToolKit method with both EKF and IMU. Nonetheless, we found in practice that performance in the visual coordination tasks was already good without the further addition of the IMU. This system has a simple setup and an easy initialization phase, works at the same camera frame acquisition rate (i.e., 30 Hz), and returns accurate camera pose estimations relative to the desired world reference frame (note that the NICP and PTAM methods provide instead pose estimates expressed with respect to their initial frame).

IV. Motion Control

We present now the design of a kinematic control law that handles the visual coordination task introduced in Sec. II, including the relaxed definition of the pointing task. Two strategies are pursued: the first one is by augmentation of the positional task ($m_P = 3$) with the relaxed pointing task ($m_{rp} = 1$), as detailed in Sec. IV-A; the second is through a suitable projection of the relaxed pointing task into the null space of the positional one (Sec. IV-B).

A. Task Augmentation (TA)

Consider first a desired task $\boldsymbol{x}_d(t) \in \mathbb{R}^3$ defined for the robot end-effector position,

$$\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q}) \quad \Rightarrow \quad \dot{\boldsymbol{x}} = \boldsymbol{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the robot configuration, $\boldsymbol{k}(.)$ is the direct kinematics, and $\boldsymbol{J}_P = \partial \boldsymbol{k}/\partial \boldsymbol{q}$ is the $3 \times n$ Jacobian matrix for this task. The positional task is executed in nominal conditions by choosing a $\dot{\boldsymbol{q}}$ satisfying $\boldsymbol{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}_d$. The position error is defined as $\boldsymbol{e}_P = \boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}) \in \mathbb{R}^3$.

The relaxed pointing task is specified by a constant relative angle $\alpha_d \in \mathbb{R}$ between a unit vector $\boldsymbol{z}_d(t)$ and the $\boldsymbol{z}_e(\boldsymbol{q})$ axis of the frame attached to the robot end-effector(the third column of the rotation matrix $\boldsymbol{R}_e(\boldsymbol{q})$ relative to the world frame). We have

$$f(\boldsymbol{q}) = \boldsymbol{z}_d^T \boldsymbol{z}_e(\boldsymbol{q}) = \cos\alpha. \tag{2}$$

For a constant desired relative angle $\alpha_d > 0$, let $f_d = \cos\alpha_d$. The relaxed pointing error is defined as $e_{rp} = f_d - f(\boldsymbol{q}) \in \mathbb{R}$.

We derive next the Jacobian associated to the relaxed pointing task. Since α is assumed to be constant, differentiating eq. (2) yields

$$\frac{df(\boldsymbol{q})}{dt} = \frac{d\boldsymbol{z}_d}{dt}^T \boldsymbol{z}_e(\boldsymbol{q}) + \boldsymbol{z}_d^T \frac{d\boldsymbol{z}_e(\boldsymbol{q})}{dt} = 0. \tag{3}$$

Being the time derivative of $\boldsymbol{R}_e(\boldsymbol{q})$ [16]

$$\frac{d\boldsymbol{R}_e(\boldsymbol{q})}{dt} = \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}_e(\boldsymbol{q}),$$

where $\boldsymbol{S}$ is the 3×3 skew-symmetric matrix representing the vector ($\times$) product and $\boldsymbol{\omega}$ denotes the angular velocity of frame $\boldsymbol{R}_e(\boldsymbol{q})$, we have

$$\frac{d\boldsymbol{z}_e(\boldsymbol{q})}{dt} = -\boldsymbol{S}(\boldsymbol{z}_e(\boldsymbol{q}))\boldsymbol{\omega} = \boldsymbol{S}^T(\boldsymbol{z}_e(\boldsymbol{q}))\, \boldsymbol{J}_O(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{4}$$

where $\boldsymbol{J}_O(\boldsymbol{q})$ is the $3 \times n$ Jacobian that maps the joint velocity $\dot{\boldsymbol{q}}$ to the end-effector angular velocity $\boldsymbol{\omega}$. Substituting (4) in (3), we obtain

$$\frac{d\boldsymbol{z}_d}{dt}^T \boldsymbol{z}_e(\boldsymbol{q}) + \boldsymbol{J}_{rp}(\boldsymbol{q})\dot{\boldsymbol{q}} = 0, \tag{5}$$

where

$$\boldsymbol{J}_{rp}(\boldsymbol{q}) = \boldsymbol{z}_d^T\, \boldsymbol{S}^T(\boldsymbol{z}_e(\boldsymbol{q}))\, \boldsymbol{J}_O(\boldsymbol{q}) \tag{6}$$

is the $1 \times n$ Jacobian matrix associated to the task defined through (2). From (5), our relaxed pointing orientation task is executed in nominal conditions by choosing a $\dot{\boldsymbol{q}}$ that satisfies

$$\boldsymbol{J}_{rp}(\boldsymbol{q})\dot{\boldsymbol{q}} = -\dot{\boldsymbol{z}}_d^T(t)\, \boldsymbol{z}_e(\boldsymbol{q}) = \dot{r}_d. \tag{7}$$

The complete visual coordination task can be realized using Task Augmentation [10], namely by stacking the Jacobians $\boldsymbol{J}_P$ in (1) and $\boldsymbol{J}_{rp}$ in (7). In nominal conditions, a solution can be obtained by choosing a joint velocity $\dot{\boldsymbol{q}}$ that satisfies

$$\dot{\boldsymbol{x}}_{A,d} = \begin{pmatrix} \dot{\boldsymbol{x}}_d \\ \dot{r}_d \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_P(\boldsymbol{q}) \\ \boldsymbol{J}_{rp}(\boldsymbol{q}) \end{pmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{8}$$

where $\boldsymbol{J}_A(\boldsymbol{q})$ is the $4 \times n$ Augmented Jacobian. Assuming that $n > 4 = m$, i.e., the robot is redundant for the augmented task, and that we need to react by feedback to possible positioning error $\boldsymbol{e}_P$ and/or relaxed pointing error e_{rp} that may arise during task execution, the final control law is defined as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{\#}(\boldsymbol{q}) \left(\begin{pmatrix} \dot{\boldsymbol{x}}_d \\ \dot{r}_d \end{pmatrix} + \begin{pmatrix} \boldsymbol{K}_P(\boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q})) \\ k_{rp}(f_d - f(\boldsymbol{q})) \end{pmatrix} \right) \tag{9}$$

where $\boldsymbol{J}_A^{\#}$ is the pseudoinverse of the Jacobian $\boldsymbol{J}_A$ in (8), $\boldsymbol{K}_P > 0$ is a 3×3 diagonal gain matrix, and $k_{rp} > 0$ is a scalar gain. Typically, the 3D motion of the human/camera is not known in advance, and so $\dot{\boldsymbol{x}}_d$ and $\dot{r}_d$ in (9) will not be available to the controller. In practice, these reference velocities are set to zero and robot motion will be driven only by the two errors $\boldsymbol{e}_P$ and e_{rp}. Indeed, it is also possible to obtain an online prediction of the camera motion (e.g., based on an EKF method, as done in [17]), and include this as the nominal feedforward term in the control law (9).

To obtain the pseudoinverse in (9), one needs to compute the Singular Value Decomposition (SVD) [18] of the Jacobian, $\boldsymbol{J}_A = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U} = \mathrm{col}\{\boldsymbol{u}_i\}$ and $\boldsymbol{V} = \mathrm{col}\{\boldsymbol{v}_j\}$ are, respectively, $m \times m$ and $n \times n$ unitary matrices, and $\boldsymbol{\Sigma}$ is a $m \times n$ block matrix, with a leading $m \times m$ diagonal block containing the singular values $\sigma_i \geq 0$ ($i = 1, \ldots, m$) of $\boldsymbol{J}_A$ in decreasing order ($\sigma_h \geq \sigma_k$ for $h < k$), followed by $n - m$ zero columns. It is

$$\boldsymbol{J}_A^{\#} = \boldsymbol{V}\boldsymbol{\Sigma}^{\#}\boldsymbol{U}^T = \sum_{i=1}^{\rho} \frac{1}{\sigma_i}\boldsymbol{v}_i\boldsymbol{u}_i^T, \tag{10}$$

where $\rho \leq m = 4$ is the rank of $\boldsymbol{J}_A$. When the smallest singular value(s) becomes too small (i.e., close to singularities), large joint velocities are being generated. This drawback was addressed in our implementation by the Damped Least Squares (DLS) technique [10], replacing in (10)

$$\frac{1}{\sigma_i} \rightarrow \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \quad \Rightarrow \quad \boldsymbol{J}_A^{\#} \rightarrow \boldsymbol{J}_A^{\mathrm{DLS}}. \tag{11}$$

with

$$\lambda^2 = \begin{cases} 0, & \text{when } \sigma_i \geq \epsilon, \\ \Big(1 - (\sigma_i/\epsilon)^2\Big)\lambda_{max}^2, & \text{otherwise.} \end{cases}$$

The small parameter $\epsilon \geq 0$ monitors the singular values σ_i and defines the range of the damping action, while $\lambda_{max}^2 > 0$ is the largest damping factor used at singularities.

B. Projected Gradient (PG)

In the second control strategy, we pursue a null-space design method. Considering the positional tracking task defined through (1) as the one with highest priority, we accommodate the relaxed point task into a joint velocity $\dot{\boldsymbol{q}}_0 \in \mathbb{R}^n$ which is then projected in the null space of the high-priority task, so as not to affect its execution in case of conflicts. Thus, we define

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_P^{\#}(\boldsymbol{q})\left(\dot{\boldsymbol{x}}_d + \boldsymbol{K}_P \boldsymbol{e}_P\right) + \boldsymbol{P}(\boldsymbol{q})\dot{\boldsymbol{q}}_0 \qquad (12)$$

where $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{J}_P^{\#}\boldsymbol{J}_P$ is the $n \times n$ orthogonal projector in the null space of $\boldsymbol{J}_P$ and $\boldsymbol{I}$ is the identity matrix of the same size. To execute also the relaxed pointing task, a Projected Gradient technique [10] is followed, choosing vector $\dot{\boldsymbol{q}}_0$ as

$$\dot{\boldsymbol{q}}_0 = -k_g \nabla_{\boldsymbol{q}} H(\boldsymbol{q}), \qquad (13)$$

with

$$H(\boldsymbol{q}) = \frac{1}{2}\, e_{rp}^2 = \frac{1}{2}\left(\cos\alpha_d - \boldsymbol{z}_d^T \boldsymbol{z}_e(\boldsymbol{q})\right)^2,$$

namely along the negative gradient of the squared norm of the error e_{rp}, taken as objective function $H(\boldsymbol{q}) \geq 0$. The scalar gain $k_g > 0$ in (13) is used to shape the convergence rate, and plays a very similar role as k_{rp} in (9).

C. Task Limitations

The visual coordination task is summarized with further details in Fig. 6. The tip of the robot should follow a point $\boldsymbol{x}_d$, attached to the moving camera, which is displaced by $h > 0$ along the camera frame axis $-\boldsymbol{y}_c$ and located at a distance $d > 0$ along its $\boldsymbol{z}_c$ axis. The end-effector orientation of the robot should be such that its approach axis $\boldsymbol{z}_e$ is kept at an angle α_d from the desired direction $\boldsymbol{z}_d$, which is ideally pointing at the human eyes.

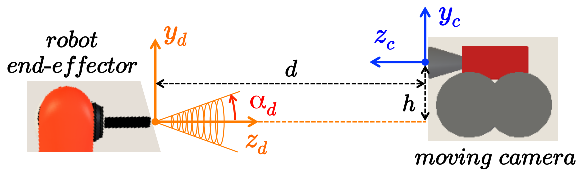


Fig. 6: Frames and parameter definitions for the visual coordination task. Here, $\boldsymbol{z}_e = \boldsymbol{z}_d$ yielding $\alpha = 0 \neq \alpha_d$.

During the execution of the complete visual coordination task, some limits may be reached as well as task singularities encountered. While singularities are handled properly by the DLS method in (11), the occurrence of task limits deserve a special treatment. We define the limits of a coordination task by a sphere with center located at the (spherical) shoulder of the considered KUKA LWR robot, and having a radius $R = 1$ m, see Fig. 7. If the camera position is outside the sphere and an intersection exists between its line of sight and the sphere, then the desired end-effector position will be relocated accordingly at the intersection point. Otherwise, if there is no intersection, the robot will be commanded using the last computed $\boldsymbol{x}_d$ and f_d, reduce then the residual errors to zero, and finally remaining at rest. Another situation which is beyond the task limits occurs when the camera is inside the sphere but looking to the outside of the robot workspace. Also in this case the robot will be commanded as before. As soon as the position and pointing direction of the camera become again feasible for the task, motion control is resumed and new detected errors are recovered by the laws (9) or (12).

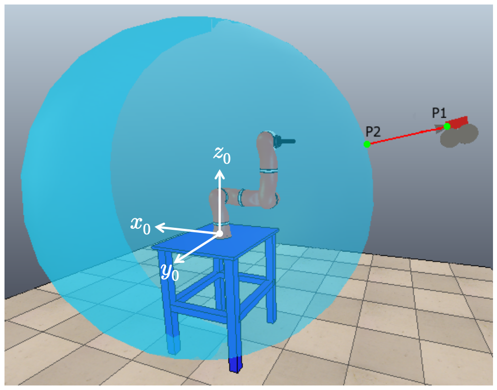


Fig. 7: The blue sphere represents a limit for the visual coordination task. When the camera is outside the sphere in the position P1, the projected position P2 on the boundary of the sphere will be used as reference for motion control.

V. Results

A. Simulations

Several simulations have been run in a ROS environment, integrated with the robot simulator V-REP, using the motion control law (9) or (12). Figure 8 shows in the V-REP scenario some typical robot configurations obtained for different camera poses.

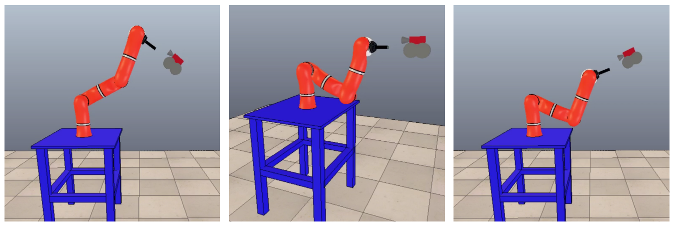


Fig. 8: Representative robot behaviors achieved during the visual coordination task.

In all the numerical tests, we assumed an ideal localization. The two offsets used to determine the desired position $\boldsymbol{x}_d$ and the constant relative angle for the relaxed pointing task (see Fig. 6) were chosen as

$$h = 0.05 \text{ [m]}, \qquad d = 0.2 \text{ [m]}, \qquad \alpha_d = 5^\circ. \qquad (14)$$

The gains in the control law (9) were chosen as

$$\boldsymbol{K}_P = \operatorname{diag}\{20, 30, 30\}, \qquad k_{rp} = 20.$$

Moreover, the feedforward terms in (9) were absent ($\dot{\boldsymbol{x}}_d = \boldsymbol{0}$, $\dot{r}_d = 0$), since the camera motion is assumed to be unknown.

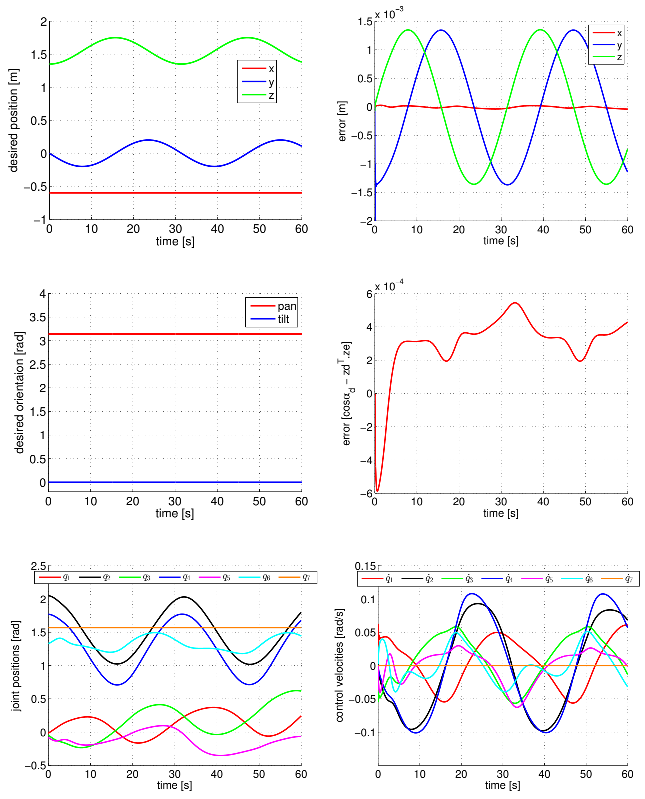


Fig. 9: Motion control with the TA method in the first simulated task: (top) desired end-effector position $\boldsymbol{x}_d$ and position error $\boldsymbol{e}_P$; (center) desired pointing direction $\boldsymbol{z}_d$, represented by its pan and tilt angles, and relaxed pointing error e_{rp}; (bottom) joint positions and commanded velocities.

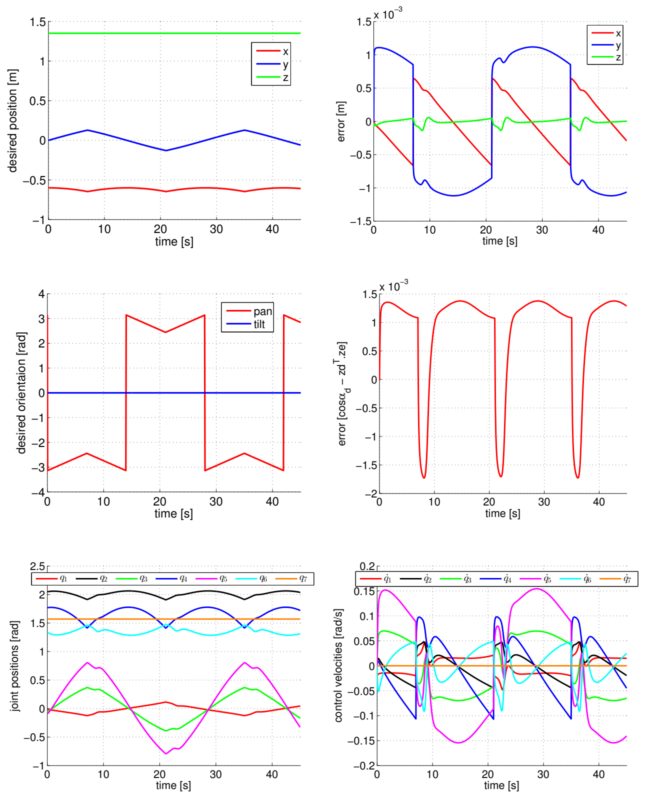


Fig. 10: Motion control with the TA method in the second simulated task. Quantities are the same reported in Fig. 9.

In Figs. 9–10, we report the results obtained for two simple camera motions. In both cases, the robot initial configuration is matched with the initial desired task, i.e., at time $t = 0$, the errors are $\boldsymbol{e}_P(0) = \mathbf{0}$ and $e_{rp}(0) = 0$.

In the first simulation, the camera position changes continuously, tracing a circle in the vertical plane at $\boldsymbol{x}_0 = -0.8$ m (parallel to $(\boldsymbol{y}_0, \boldsymbol{z}_0)$), while the pointing direction is kept constant and horizontal at the value $\boldsymbol{z}_d = (-1, 0, 0)$. Since the motion is relatively slow, both the maximum of the pointing error ($e_{rp,max} = 5.4 \cdot 10^{-4}$) and the maximum norm of the position error ($\|\boldsymbol{e}_P\|_{max} = 2 \cdot 10^{-3}$) in Fig. 9 are very small. In the second simulation, the camera center is kept fixed while $\boldsymbol{z}_d$ oscillates periodically in a horizontal plane around the vertical axis. As a result, the reference position of the displaced target point $\boldsymbol{x}_d$ is changing, as shown in Fig. 10. The peaks or discontinuities in the errors occur when the x and y coordinates of $\boldsymbol{x}_d$ are inverting motion, but the errors remain always small. The results obtained using the PG control method are very similar to those shown with TA control, and are thus not reported.

B. Experiment

Experiments were conducted with a KUKA LWR IV manipulator in a ROS environment, using the joint position control mode of the FRI with a sampling time $t_s = 5$ ms. An Intel core i5 @2.5GHz×4 laptop was used under 64-bit Ubuntu. For the online localization of the Kinect camera[1], three markers were placed on the robot supporting table and the ARToolKit library with an additional EKF was used. The camera is being held by an operator who is moving around in the workspace tracing an arbitrary, a priori unknown path, see Fig. 11. The offset data h and d, and the relative angle α_d were chosen as in (14). Task augmentation was used to

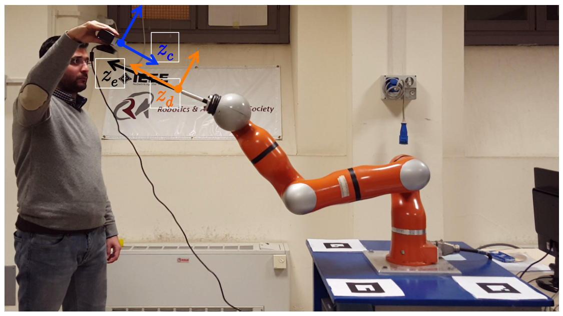


Fig. 11: Snapshot of the experiment. The camera is simply held by hand and moved around, but frames and offsets mimic the situation of a camera mounted on the human head. Three markers are used for continuous camera localization. Depending on the camera pose, the marker with the highest confidence factor is used.

[1]For the Microsoft Kinect, we used the libfreenect-based ROS driver, where the camera parameters can be found and edited.

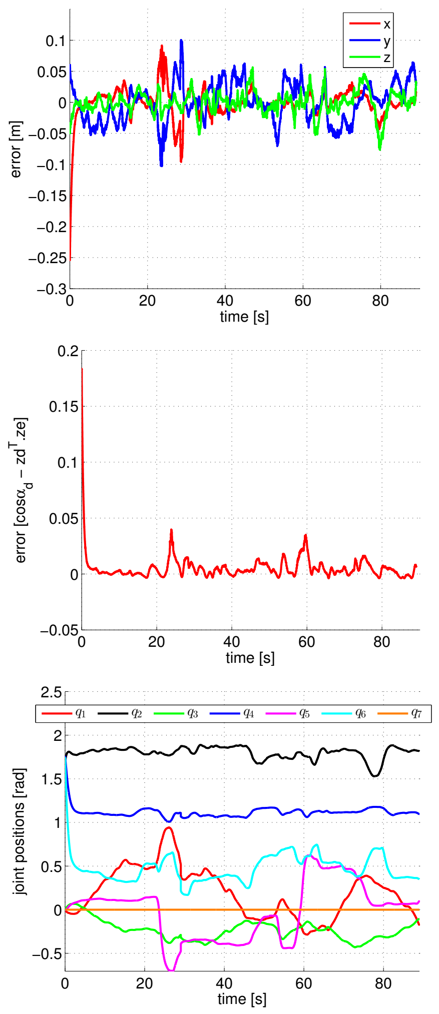


Fig. 12: Experimental results with the KUKA LWR IV: (top) position error; (center) relaxed pointing error; (bottom) evolution of the joints.

control the robot motion, with (cautious) gains $\boldsymbol{K}_P = 1.5 \cdot \boldsymbol{I}$ and $k_{rp} = 1.5$, and no feedforward.

One representative experiment is shown in the accompanying video, and the obtained results are reported in Fig. 12. The error peaks on position and pointing angle are related to fast transients in the camera motion, leading to larger actual differences with respect to the desired visual coordination task. Nonetheless, once the initial mismatching error is recovered, the maximum of the (non-dimensional and normalized) pointing error was $e_{rp,max} \simeq 0.04$. Similarly, $\|\boldsymbol{e}_P\|_{max} \simeq 0.14$ m.

VI. Conclusions

In the framework of contactless human-robot collaboration, we have defined and validated through simulations and experiments a generic visual coordination task. A camera attached to a moving human is efficiently localized online by an enhanced method that combines the ARToolkit library and data processing by EKF. Coordinated motion of the robot with the human is obtained via a special task definition, which involves three positional variables and only one angular component, and thanks to kinematic control schemes (based on task augmentation or on the projected gradient) that handle robot redundancy.

As future work, we plan to use the full RGB-D sensor, including depth, and develop real-time collision avoidance schemes as in [3], so as to guarantee coexistence in human-robot interaction with a moving sensor. Also, we could exploit better in the control law the larger null space associated to the special definition of the visual coordination task.

References

[1] A. De Luca and F. Flacco, "Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration," in *Proc. 4th IEEE Int. Conf. on Biomedical Robotics and Biomechatronics*, 2012, pp. 288–295.

[2] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 3356–3363.

[3] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach for evaluating distance to objects – with application to human-robot collision avoidance," *J. of Intelligent & Robotic Systems*, vol. 80, Suppl. 1, pp. 7–22, 2015.

[4] E. Magrini, F. Flacco, and A. De Luca, "Control of generalized contact motion and force in physical human-robot interaction," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 2298–2304.

[5] O. Rogalla, M. Ehrenmann, R. Zollner, R. Becher, and R. Dillmann, "Using gesture and speech control for commanding a robot assistant," in *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication*, 2002, pp. 454–459.

[6] C. Nehaniv, K. Dautenhahn, J. Kubacki, M. Haegele, C. Parlitz, and R. Alami, "A methodological approach relating the classification of gesture to identification of human intent in the context of human-robot interaction," in *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication*, 2005, pp. 371–377.

[7] J. Mainprice, E. A. Sisbot, T. Siméon, and R. Alami, "Planning safe and legible hand-over motions for human-robot interaction," in *Proc. IARP Work. on Technical Challenges for Dependable Robots in Human Environments*, 2010.

[8] M. Svenstrup, S. Tranberg, H. J. Andersen, and T. Bak, "Pose estimation and adaptive robot behaviour for human-robot interaction," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 3571–3576.

[9] L. Villani, A. De Santis, V. Lippiello, and B. Siciliano, "Human-aware interaction control of robot manipulators based on force and vision," in *Proc. 7th Work. on Robot Motion Control*. Lecture Notes in Control and Information Sciences, vol. 396, Springer, 2009, pp. 209–225.

[10] S. Chiaverini, G. Oriolo, and I. Walker, "Kinematically redundant manipulators," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 245–268.

[11] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. 9th IEEE Int. Conf. on Computer Vision*, 2003, pp. 1403–1410.

[12] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, no. 12, pp. 239–256, 1992.

[13] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 742–749.

[14] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proc. 6th IEEE/ACM Int. Symp. on Mixed and Augmented Reality*, 2007, pp. 225–234.

[15] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proc. 2nd IEEE/ACM Int. Work. on Augmented Reality*, 1999, pp. 85–94.

[16] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, planning and control*. Springer, 2010.

[17] G. Milighetti, L. Vallone, and A. De Luca, "Adaptive predictive gaze control of a redundant humanoid robot head," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 3192–3198.

[18] G. Golub and C. Van Loan, *Matrix Computations*. Johns Hopkins Univ. Press, 1996.