Citation for the original published paper:

Båberg, F., Wang, Y., Caccamo, S., Ögren, P. (2016)
Adaptive object centered teleoperation control of a mobile manipulator.
In: *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 455-461).
Proceedings - IEEE International Conference on Robotics and Automation
http://dx.doi.org/10.1109/ICRA.2016.7487166

Adaptive Object Centered
Teleoperation Control of a Mobile Manipulator

Fredrik Båberg, Yuquan Wang, Sergio Caccamo, Petter Ögren

***Abstract*— Teleoperation of a mobile robot manipulating and exploring an object shares many similarities with the manipulation of virtual objects in a 3D design software such as AutoCAD. The user interfaces are however quite different, mainly for historical reasons. In this paper we aim to change that, and draw inspiration from the 3D design community to propose a teleoperation interface control mode that is identical to the ones being used to locally navigate the virtual viewpoint of most Computer Aided Design (CAD) softwares.**

The proposed mobile manipulator control framework thus allows the user to focus on the 3D objects being manipulated, using control modes such as *orbit object* and *pan object*, supported by data from the wrist mounted RGB-D sensor. The gripper of the robot performs the desired motions relative to the object, while the manipulator arm and base moves in a way that realizes the desired gripper motions. The system redundancies are exploited in order to take additional constraints, such as obstacle avoidance, into account, using a constraint based programming framework.

***Index Terms*— Virtual object, mobile manipulation, teleoperation**

I. INTRODUCTION

Teleoperated mobile robots equipped with manipulators are expected to play key roles in future *Search and Rescue* and *Explosive Ordnance Disposal* operations. In these applications, robots are sent to places where it is not safe for humans to go, but difficult tasks still have to be carried out.

It is well known that robot teleoperation is a demanding task [1], and a lot of research is currently aimed to improve performance and reduce operator workload in these safety critical applications.

It has been noted that robot teleoperation has many similarities with playing first person perspective computer games [2]. In both cases a human is controlling an entity that is moving around in a remote environment trying to achieve a specific task. These similarities have been used to improve many parts of the teleoperation, from using gamepads for input, to designing control modes and the presentation of video streams and other sensing modalities.

In this paper, we draw inspiration from another area of virtual reality. Instead of computer games, we look at the interfaces of *3D design software* such as Autodesk AutoCAD, SolidWorks, V-REP[1] and Gazebo[1]. These software tools are used by engineers and architects to make 3D drawings and designs. When manipulating objects, the users navigate the virtual space using the functions *pan object* and *orbit object*, see Figure 1. These functions are the core part of an interface that is used daily by thousands of professionals and has been refined in many iterations. Therefore, there is reason to believe that the same functions would be useful when exploring and manipulating remote objects with a teleoperated mobile manipulator equipped with a RGB-D sensor[2].

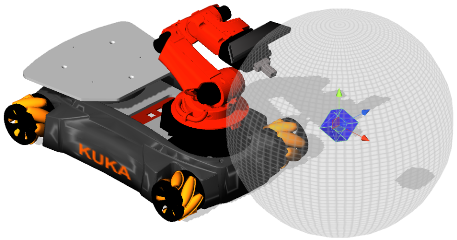

Fig. 1. Concept illustration. A mobile manipulator with a virtual sphere. The blue cube in the center of the sphere is the object to be examined. In the control mode *orbit object*, the end effector moves on the sphere while keeping the object in the center of view, when commanded left/right and up/down.

With the proposed approach, the user can choose between *pan object* and *orbit object* when controlling the robot. In both control modes, the full pose of the end effector, position and orientation is controlled using a gamepad, and the video stream from the RGB-D sensor mounted on the wrist of the end effector is shown to the user.

In *pan object*, the sensor equipped end effector moves in a so-called robot centric way, known from the literature, see e.g. [3]. A requested translation forwards results in the end effector moving on a straight line in the direction towards whatever is in the center of view, and a requested translation to the right results in a straight line to the right.

In *orbit object*, the motions are *object centric*, and relative to the object in the center of the sensor view. Also here, as for *pan object*, a requested forward translation results in a

---

The authors are with the Computer Vision and Active Perception Lab., Centre for Autonomous Systems, School of Computer Science and Communication, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden. e-mail: `{fbaberg|yuquan|caccamo|petter}@kth.se`

[1] The last two examples are actually robot simulators, but this functionality concerns creating 3D environments

[2] Red Green Blue plus Depth, image including depth/distance information. For instance Microsoft Kinect, Intel RealSense, PrimeSense.

straight line motion towards the object in the center of the view. A requested translation to the right on the other hand, results in a circular arc trajectory orbiting the object in center of the view, and keeping that object in the center of the view by a corresponding rotation, see Figure 1. The radius of the orbit motion is given by the current distance to the center of the object, which is estimated using the RGB-D sensor.

The two control modes described above complement each other, and are often used in an alternating fashion. However, as *pan object* is equivalent with robot centric control, [3], we focus this work on *orbit object* which is new to the robotics community.

In the design applications, *orbit object* is useful for exploring an object, and moving into a viewpoint that allow you to add or remove details. In the robot teleoperation application, we believe that *orbit object* would be useful for exploring objects, getting good camera views from all sides. It would also be useful when gathering 3D data from the RGB-D sensor, in order to create a high quality 3D-model. Finally, it would be convenient when deciding on the appropriate grasp point for lifting an object, or operating a door handle.

The approach presented in this paper realizes the *orbit object* control mode using constraint based programming. To realize the appropriate motion of the end effector, the configuration of the complete mobile manipulator must be taken into account. Sometimes it is best to move only the arm, but sometimes the mobile base has to be moved to increase the range of the arm, while simultaneously taking obstacles and internal singularities into account. All this has to happen automatically, enabling the user to focus on the task at hand, which is being carried out by moving the end effector relative to the object of interest.

The contribution of this paper is that we show how to realize the control mode *orbit object* in a teleoperated mobile manipulator. To the best of our knowledge, this has not been done before. We also show how to incorporate avoidance of obstacles into the framework using constraint based programming.

The structure of the paper is as follows. First, Section II describes related work. Then, in Section III we will provide some notation and definitions, and formulate the problem, before proposing a solution in Section IV. The solution is verified with experiments in Section V, and finally conclusions can be found in Section VI.

II. RELATED WORK

As the proposed approach involves teleoperation of a mobile manipulator, we will first discuss work on teleoperation of mobile robots, and then manipulators.

Within the area of search and rescue robotics there has been a lot of work on teleoperation of mobile robots, and a nice overview of the problems involved can be found in [1] and [4]. While [1] describes the domain in detail, [4] suggest possible improvements in terms of multimodal feedback, such as using combinations of video, audio, and haptics.

In a study based on experiences from the AAAI Robot Rescue Competitions in 2002-2004 [5], the authors noticed an evolution over time, towards a large single interface, with a large percentage of the screen dedicated to video.

The idea of supporting user situation awareness with a virtual 3D rendering of the robot and its surroundings was explored in [6] and [7] and the use of multi-touch Operator Control Units (OCUs) including fusion of sensor information to lower the operator's cognitive load was investigated in [8].

In [9] the authors identify seven fundamental problems in OCU design, propose a solution focussing on sensor data presentation, and present results from end-user evaluations.

The proposed paper differs from the work above in that none of the above consider the actual control layer of the OCU, instead they focus on how information is presented to the operator.

Within the area of mobile robot control, a lot of inspiration has been drawn from similarities between computer games, and robot teleoperation, and [2] is an early study on this topic. There it is argued that Video Game Based Frameworks (VGBF) are very useful for both evaluating existing interfaces and inspiring the design of new ones. The authors then go on to make a detailed categorization of input and output devises as well as methods used in different games and discuss different combinations of real video streams and rendered images of the vehicle surroundings.

One way of using inspiration from computer games was presented in [10], where the classical robot control mode of *Tank Control* was replaced with *Free Look Control* which is used in many computer games.

In our work, we draw inspiration from virtual interfaces, but our inspiration comes not from computer games, but from professional modeling tools such as AutoCAD.

There has also been a lot of studies into the area of teleoperation for manipulation. The importance of different reference frames, i.e. robot centric or view centric, was explored in [3], where the author propose an Ecological interface design that aims to make relationships in the environment perceptually evident to the user, in order to minimize the effort needed for understanding those relationships.

The use of smartphones or tablets to control a manipulator was investigated in [11], where the operator could either modify the target position of the end effector in the workspace, or use the high level skill of autonomous grasping.

The effect of stereoscopic displays on task performance and cognitive workload was investigated in [12], and performance on different autonomy levels was studied in [13]. The levels included direct control, waypoint control, indication of general grasps area, and completely autonomous grasping.

Performing manipulation with user input in terms of 2D click and drag input from a mouse was explored in [14]. There, five different strategies were investigated, including joint space control, cartesian space control, and 3 versions of obstacle avoidance based on reactive control, filtered prediction and motion planning.

Teleoperation of a 8 DoF mobile manipulator using a 6

DoF joystick was investigated in [15]. The authors propose a control approach where the user controls the gripper pose, while the mobile base adapts and follows the gripper when possible and needed, to avoid over extending the arm.

The approach proposed in this paper differs from all the above in that the control mode is neither robot centric nor world centric, but object centric. In the *orbit object* control mode, a commanded motion to the right results in moving right with respect to the gripper, but keeping a constant distance to the object. This is motivated by an argument similar to the ones suggesting inspiration from computer game interfaces [2], but this time the inspiration comes from the professional 3D design community.

Motions can be constrained through virtual fixtures [16], and the approach used in this paper can be seen as such in the sense that regions can be restricted through constraints. However our implementation does not use force sensors for feedback, which is the case in for instance [17].

Introducing different constraints for restricting motion could introduce conflicts, in which case it could be necessary to prioritize [18]. In this paper we do not explicitly consider priorities, however by changing weights and introducing slack variables this could be considered.

III. PROBLEM FORMULATION

In this section, we describe *orbit object* in more detail, and establish the notation used in the paper.

Boldface will indicate vectors, and the indices w, b, a, e, o denote **w**orld, robot **b**ase, **a**rm base, **e**nd effector and **o**bject respectively. The frames can be seen in Figures 2 and 3.

$\mathbf{q}$ - joint positions
$\mathbf{p}^i_j$ - position of object j in frame i
$r(t)$ - distance between end effector and object
J - Spatial Jacobian
f_i - constraints
$Ad_{g_{xy}}$ - Adjoint transformation from x to y
$\mathbf{e}^i_j$ - unit vector j in frame i
$\dot{\mathbf{p}}_{ij}$ - velocity of object j with respect to object i (in frame i).
R^i_j - Rotation matrix of object j in relation to frame i.
$\mathbf{v}^e_d$ - desired end effector velocity (user input)
ω^e_d - desired end effector angular velocity (user input)
$\mathbf{u}$ - joint velocity for arm+base, from optimization problem.
I_e, I_{ie} - Set of indices for equalities and inequalities.

A dot above a symbol indicates differentiation with respect to time, e.g. $\dot{\mathbf{q}}$ denotes the joint velocities. Superscript indicates which frame is used. We now define the control mode.

Orbit Object To move on the surface of a sphere, centered on the object, with a fixed radius. In Figure 2 this would correspond to moving along the surface of the sphere, with constant radius $r(t)$, and with x-axis at $\{e\}$ aligned with the vector between $\{e\}$ and $\{o\}$, i.e. always facing the centre of the sphere.

Given the definition above, we aim to solve the following problem.

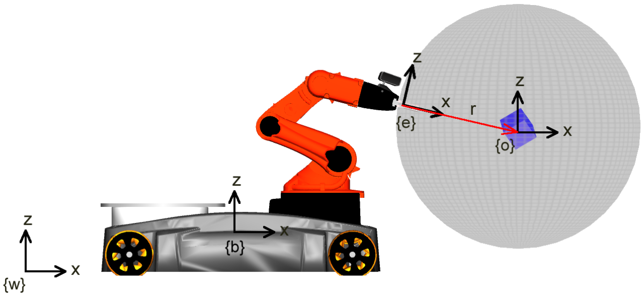

Fig. 2. Illustration of frames and points defined, side view.

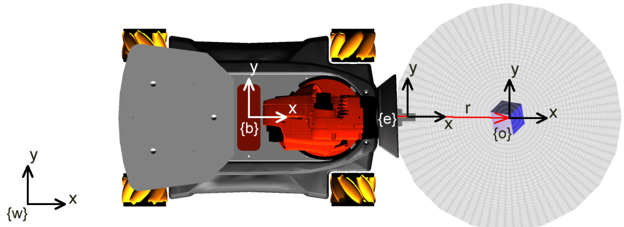

Fig. 3. Illustration of frames and points defined, top view.

Problem 3.1: Implement the control mode above, while avoiding collisions.

We will now describe the proposed solution.

IV. PROPOSED SOLUTION

We propose to use a Constraint Based Programming (CBP) framework in order to solve Problem 3.1. Following the approach presented in [19] we first describe the problem we want to solve, and then state a reactive algorithm where a convex quadratic programming (QP) problem is solved in each timestep, taking the user input and the current state of all constraints into account.

Problem 4.1: Given a time interval $[t_0, t_f]$, initial state $q(t_0) = q_0$ and a control system

$$\dot{q} = h(q, u),$$

where $q \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$. Let us formulate the control objective in terms of a set of functions $f_i : \mathbb{R}^n \to \mathbb{R}$ and bounds $b_i \in \mathbb{R}$, $i \in I \subset \mathbb{N}$ as follows

$$\min_{u(\cdot)} \quad f_j(q(t_f), t_f), \ j \in I \tag{1}$$

$$\text{(s.t.)} \quad f_i(q(t), t) \leq b_i, \ \forall i \in I_{ie}, \ t > t_0 \tag{2}$$

$$f_i(q(t), t) = b_i, \ \forall i \in I_e, \ t > t_0 \tag{3}$$

where we assume that the constraints are satisfied at t_0, i.e. $f_i(q_0, t_0) \leq b_i$ for all $i \in I_{ie}$ and $f_i(q_0, t_0) = b_i$ for all $i \in I_e$ and $I_{ie}, I_e \subset I$.

Now, instead of addressing Problem 4.1 above directly, we look at a related problem where the constraints above are turned into feedback form, with controls moving the system back towards satisfying the constraints if they are momentarily not met due to e.g. uncertainties or disturbances. The related problem describes an online local controller, that also takes user input into account at each time step.

Problem 4.2:

$$\min_{u} \quad \dot{f}_j(q(t),u,t)+u^TQu,\ j\in I \tag{4}$$

$$\text{(s.t.)} \quad \dot{f}_i(q,u,t)\leq -k_i(f_i(q,t)-b_i),\ \forall i\in I_{ie}, \tag{5}$$

$$\dot{f}_i(q,u,t)= -k_i(f_i(q,t)-b_i),\ \forall i\in I_{e}, \tag{6}$$

where k_i are positive scalars and Q is a positive definite matrix.

First we look at the inequalities. It is clear that Equation (2) is satisfied for $t>t_0$ as long as Equation (5) is satisfied. Furthermore, in the worst case, if we have equality in Equation (5) then the bounds of Equation (2) will be exponentially approached, but not violated, with time constant $1/k_i$. Note that the bound will *only* be approached if motion in that direction corresponds to an improvement in the objective function, or is needed with respect to some other constraint.

Looking at the equalities, we also see that as long as Equations (6) are satisfied, so will (3), for $t>t_0$. Furthermore, if we have an error in the desired equality (3), then (6) will drive that error down to zero exponentially, with time constant $1/k_i$.

Then in the objective function, we know that (1) is kept small as long as its derivative $\dot{f}_j(q(t),u,t), j\in I$ is minimized. We smooth the input u by adding a quadratic regularization term u^TQu in (4), where Q is a diagonal positive-definite matrix designed to weight elements in u.

In order to address Problem 3.1 we need to provide a mapping between Problem 3.1 and 4.1. Then, we rely on the formalism above and iteratively solve Problem 4.2 in order to solve the two first ones.

Problem 3.1 can be captured in terms of the following constraints and corresponding equations.

- Keep desired distance from object, (7)
- Keep desired orientation w.r.t object, (8) and (9)
- Limit minimum end-effector altitude, (10)
- Avoid collision between robot and object, (11)
- Move according to user input. (12)

Formally, the constraints can be stated as

$$f_1 := ||\mathbf{p}_o^e||_2 = r_d \tag{7}$$

$$f_2 := \mathbf{p}_o^{e\top}\mathbf{e}_x^e - ||\mathbf{p}_o^e||_2||\mathbf{e}_x^e||_2 = 0, \tag{8}$$

$$f_3 := \mathbf{e}_z^{b\top}\mathbf{e}_y^e = 0, \tag{9}$$

$$f_4 := \mathbf{p}_e^{b\top}\mathbf{e}_z^b \geq z_{min}, \tag{10}$$

$$f_5 := ||\mathbf{p}_b^w - \mathbf{p}_{obs}^w||_2 \geq r_r, \tag{11}$$

$$f_6 := \dot{\mathbf{p}}_e^e = \mathbf{v}_d^e, \tag{12}$$

where r_d denotes desired distance between end-effector and object, given by the user. z_{min} is the minimum vertical separation of the end-effector and the robot base, r_r the minimum distance from obstacles and $\mathbf{v}_d^e$ is the desired end-effector movement given by user input. Only the movement in y- and z-direction, in the end-effector frame, is considered in constraint f_6 since the distance is given by constraint f_1. Note that for readability there is a mixture of frames used in the constraints. Also note that there are inequalities in f_4, f_5 whereas the rest are equalities, thus $I_{ie}=\{4,5\}$ and $I_e=\{1,2,3,6\}$.

Having stated the constraints we now need to provide their time derivatives in order to formulate Problem 4.2. Details of how the derivatives were obtained can be found in the Appendix. In this paper we assume both the object to be inspected and the obstacle to be stationary. In the following, J_t and J_ω denotes the translational and rotational part of the Jacobian matrix. Unless otherwise stated, the Jacobian matrix is given in the world frame, $J=[Ad_{g_{wa}}J_{arm}, J_{base}]$.

$$\frac{\partial f_1}{\partial q} = -\frac{\mathbf{p}_o^{e\top}J_t}{\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e}} \tag{13}$$

$$\frac{\partial f_2}{\partial q} = -\mathbf{p}_o^{e\top}S(R_e^w\mathbf{e}_x^e)J_\omega - (R_e^w\mathbf{e}_x^e)^\top J_t + \frac{\mathbf{p}_o^{e\top}J_t}{\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e}} \tag{14}$$

$$\frac{\partial f_3}{\partial q} = -\mathbf{e}_z^{b\top}S(R_e^b\mathbf{e}_y^e)J_\omega^b \tag{15}$$

$$\frac{\partial f_4}{\partial q} = -\mathbf{e}_z^{b\top}J_t^b \tag{16}$$

$$\frac{\partial f_5}{\partial q} = -\frac{\mathbf{p}_{obs}^{b\ \top}J_t}{\sqrt{\mathbf{p}_{obs}^{b\ \top}\mathbf{p}_{obs}^b}} \tag{17}$$

$$\frac{\partial f_6}{\partial q} = J_t^e \tag{18}$$

Putting it all together we get Problem 4.3, the *orbit object* version of Problem 4.2. In this case we let $\dot{f}_j=0$, as the youBot arm only has 5 DoFs.

Problem 4.3:

$$\text{minimize} \quad \mathbf{u}(t)^\top\mathbf{Q}\cdot\mathbf{u}(t)$$

subject to

$$\frac{\partial f_1}{\partial q}\mathbf{u}(t) = k_e(r_d - ||\mathbf{p}_o^w - \mathbf{p}_e^w||_2)$$

$$\frac{\partial f_2}{\partial q}\mathbf{u}(t) = k_e(0.0 - (\mathbf{p}_o^{e\top}R_e^w\mathbf{e}_x^e - ||\mathbf{p}_o^e||_2||R_e^w\mathbf{e}_x^e||_2))$$

$$\frac{\partial f_3}{\partial q}\mathbf{u}(t) = k_e(0.0 - (\mathbf{e}_z^{b\top}R_e^b\mathbf{e}_y^e))$$

$$\frac{\partial f_4}{\partial q}\mathbf{u}(t) \leq -k_{ie}(z_d - (\mathbf{p}_e^{w\top}\mathbf{e}_z^w))$$

$$\frac{\partial f_5}{\partial q}\mathbf{u}(t) \leq -k_{ie}(r_r - (||\mathbf{p}_b^w - \mathbf{p}_{obs}^w||_2))$$

$$\frac{\partial f_6}{\partial q}\mathbf{u}(t) = \mathbf{v}_d^e$$

where k_e, k_{ie} are weights for the equality- and inequality constraints.

V. SIMULATIONS

To illustrate the proposed approach, V-REP is used to simulate a KUKA youBot platform equipped with a youBot arm. On the sensor carrier of the arm, a kinect-like camera is mounted, providing RGB-D data.

The code is written in C++, and runs in Ubuntu 14.04 with ROS Indigo. The simulator has a scene with a youBot, equipped with an arm and a kinect camera, and an object to be examined. From V-REP the object location, expressed in the world frame, is obtained. Odometry data and joint states are provided as normal ROS topics. For repeatability, input is generated by given functions of time, but could easily be provided by user commands from a gamepad. Gurobi is used for solving the optimization problem.

The task is to examine a cube shaped object, with each side 0.2 m, using the *orbit object* control mode, as seen in Figure 4. This requires movement of both the arm and the base.

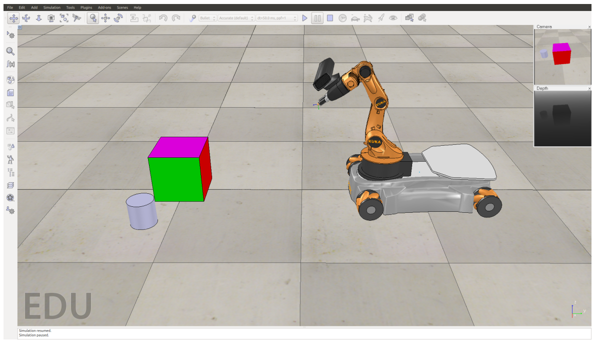

Fig. 4. A scene from V-REP, with the youBot executing the algorithm.

Running the algorithm, we get the results shown in Figures 5-18.

As illustrations, two different cases of movements will be presented. One is orbiting by moving sideways along the y-axis, while changing the desired object distance r_d (case 1), and the other is orbiting by moving upwards along the z-axis (case 2). The user inputs are given functions of time, as shown in Figures 5 and 6.

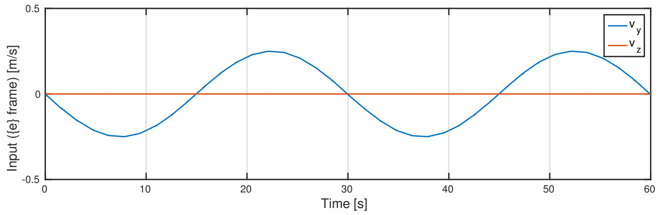

Fig. 5. Case 1: Input during orbit movement, moving in the y-direction of the camera-frame.

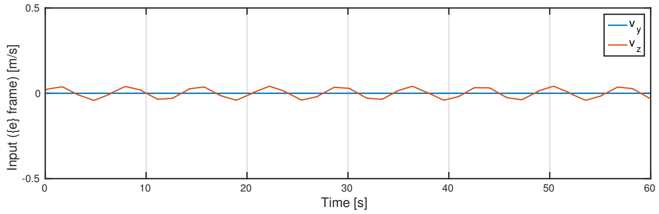

Fig. 6. Case 2: Input during orbit movement, moving in the z-direction of the camera-frame.

We will now see how well the different constraints were satisfied. The first constraint is to keep the required distance to the object, formalized in Equation (7). The corresponding results can be found in in Figures 7 and 8. Given that the approach is reactive, based on the desired user input, we cannot expect that the errors converge to zero, instead, a small remaining lag can be seen in Figure 7.

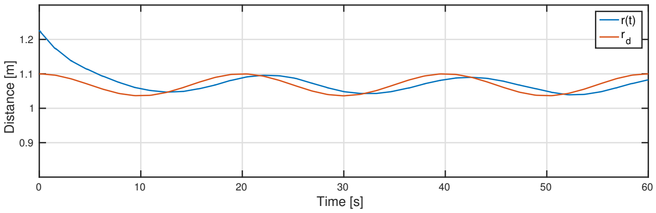

Fig. 7. Case 1: Distance between end-effector and center of object. r_d is the desired distance, and $r(t)$ is the actual distance.

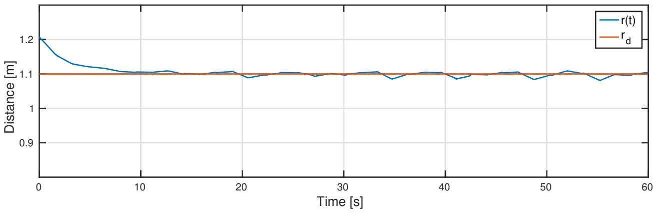

Fig. 8. Case 2: Distance between end-effector and center of object. r_d is the desired distance, and $r(t)$ is the actual distance.

The second constraint is found in Equation (8) and makes sure the object of interest is kept in the center of view. We here present the error as absolute value of an angle. At the start of the simulation, there is a significant error in end effector orientation, as can be seen in Figure 9 and 10. This is significantly reduced, but does not converge to zero. The reason is that this constraint requires motion of both base and the 5 DoF arm and the heavy base is much less precise in its motions, and there is also a modeling error in the simulator model.

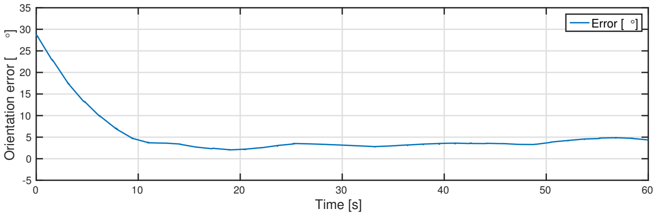

Fig. 9. Case 1: End effector orientation.

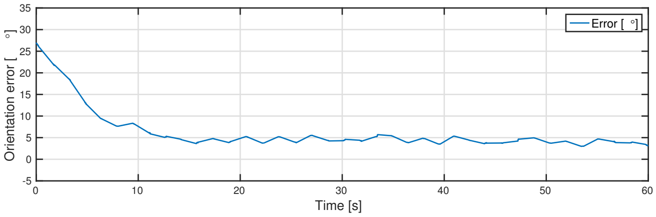

Fig. 10. Case 2: End effector orientation.

The third constraint is found in Equation (9), and makes sure the sensor is not rotating around the line of sight to the object. As can be seen in Figures 11 and 12, this constraint is kept at zero.

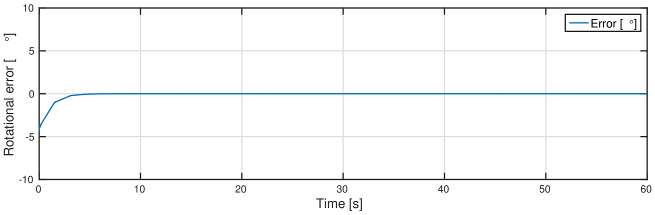

Fig. 11. Case 1: Avoiding rotation around line of sight.

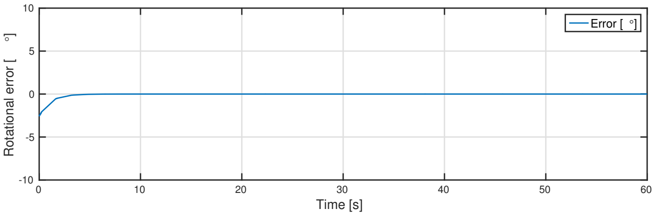

Fig. 12. Case 2: Avoiding rotation around line of sight.

The fourth constraint, found in Equation (10), makes sure that the end effector does not collide with the floor. As can be seen in Figures 13 and 14, this inequality is kept with a margin.

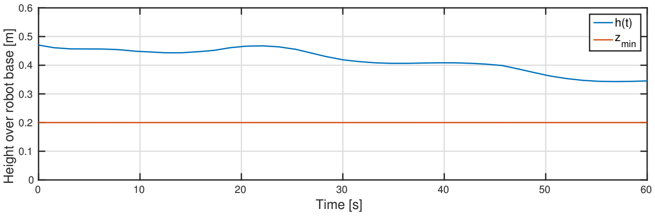

Fig. 13. Case 1: End-effector altitude over base. z_{min} is the minimum allowed height, while $h(t)$ is the actual height.

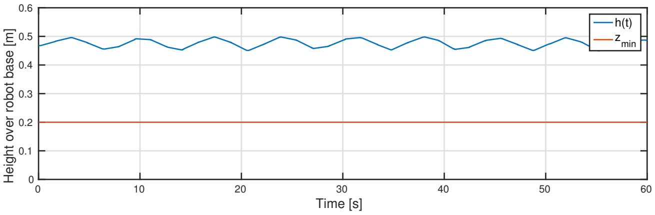

Fig. 14. Case 2: End-effector altitude over base. z_{min} is the minimum allowed height, while $h(t)$ is the actual height.

The fifth constraint, found in Equation (11), makes sure that there are no obstacle collisions. As can be seen in Figures 15 and 16, this inequality is kept with a considerable margin.

Finally, the resulting joint velocities can be found in Figures 17 and 18. As can be seen, they behave reasonably. For case 1, the sine-wave shape is clearly visible.

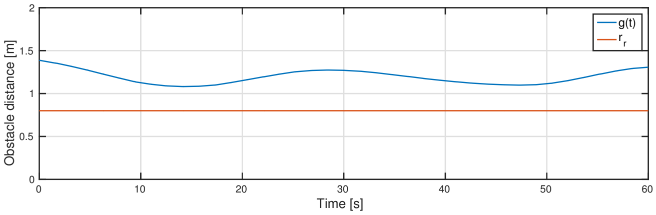

Fig. 15. Case 1: Distance from obstacle. r_r is the desired minimum distance, $g(t)$ is the distance to the obstacle.

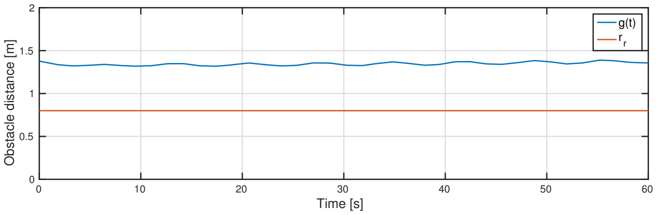

Fig. 16. Case 2: Distance from obstacle. r_r is the desired minimum distance, $g(t)$ is the distance to the obstacle.

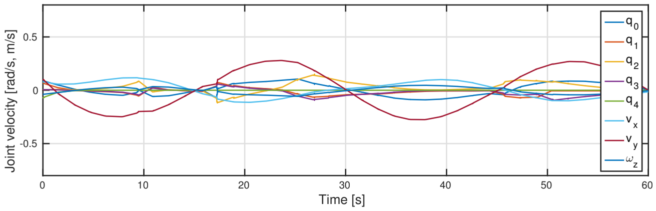

Fig. 17. Case 1: Joint velocities during orbit movement. q_0-q_4 are arm joints, v_x,v_y are translation and v_z rotation of base.

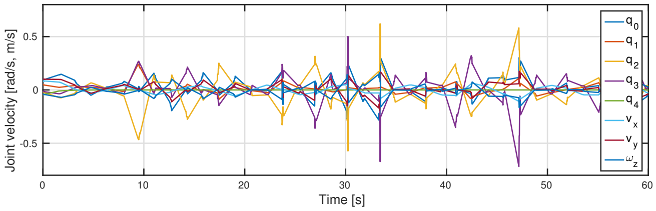

Fig. 18. Case 2: Joint velocities during height movement. q_0-q_4 are arm joints, v_x,v_y are translation and v_z rotation of base.

VI. CONCLUSIONS

In this paper we presented an approach for realizing the *orbit object* control mode on a teleoperated mobile manipulator. *Orbit object* is a key function in most 3D design softwares, enabling architects and designers to efficiently manipulate and explore virtual objects.

We believe that this *object centric* function would also provide a strong complement to the *robot centric* and *world centric* control modes described in the robot teleoperation literature.

Using a constraint based framework, we show how to implement *orbit object* on a mobile manipulator, and use V-REP simulations to illustrate the approach.

In the simulation environment the holonomic properties of the youBot has been used, specified through the Jacobian. Though not presented here, it is expected that modifying the Jacobian is sufficient to apply the algorithm to a non-holonomic platform.

APPENDIX: DERIVATIONS OF CONSTRAINTS

In this section we describe detailed derivations of the constraints.

A. Derivative of constraint f_1

Rewriting the constraint into the square root of a scalar product, the constraint follows from the product rule. We rewrite $\mathbf{p}_o^w - \mathbf{p}_e^w$ as $\mathbf{p}_o^e$, for brevity.

$$f_1 := ||\mathbf{p}_o^e||_2 = \sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e},$$

$$\frac{\partial}{\partial q}\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e} = \frac{1}{2}(\mathbf{p}_o^{e\top}\mathbf{p}_o^e)^{-1/2}\left(\frac{\partial \mathbf{p}_o^{e\top}}{\partial q}\mathbf{p}_o^e + \mathbf{p}_o^{e\top}\frac{\partial \mathbf{p}_o^e}{\partial q}\right).$$

Rewriting the sum by taking the transpose of the first term, we have that

$$\frac{\partial}{\partial q}\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e} = \frac{\mathbf{p}_o^{e\top}\frac{\partial \mathbf{p}_o^e}{\partial q}}{\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e}}.$$

We now use the fact that we are interested in the translational part, and also that the object is stationary. Thus the derivative of $\mathbf{p}_o^e$ is the Jacobi matrix of the robot base and end-effector, so we arrive at

$$\frac{\partial f_1}{\partial q} = -\frac{\mathbf{p}_o^{e\top} J_t}{\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e}}.$$

B. Derivative of constraint f_2

By similar calculations as above, we arrive at

$$\frac{\partial f_2}{\partial q} = \frac{\partial \mathbf{p}_o^e}{\partial q}^{\top}\mathbf{e}_x^e + \mathbf{p}_o^{e\top}\frac{\partial \mathbf{e}_x^e}{\partial q} - \frac{\mathbf{p}_o^{e\top}\frac{\partial \mathbf{p}_o^e}{\partial q}}{\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e}},$$

where $||\mathbf{e}_z^e||$ and it's derivative is one which simplified the last part of the expression. The final piece needed for this derivative is an expression for $\frac{\partial \mathbf{e}_x^e}{\partial q}$, which can be obtained using the skew-symmetric matrix [20][Ch. 4], to arrive at

$$\frac{\partial e_x^e}{\partial q} = -S(R_e^w e_x^e)J_\omega^b.$$

This leads to the derivative

$$\frac{\partial f_2}{\partial q} = -(R_e^w \mathbf{e}_x^e)^\top J_t - \mathbf{p}_o^{e\top} S(R_e^w \mathbf{e}_x^e) J_\omega + \frac{\mathbf{p}_o^{e\top} J_t}{\sqrt{\mathbf{p}_o^{e\top}\mathbf{p}_o^e}}.$$

C. Derivative of constraint f_3-f_6

Constraints f_3-f_6 are derived similar to f_1-f_2, and thus we omit the details.

ACKNOWLEDGMENT

The authors gratefully acknowledge funding under the European Union's seventh framework program, under grant agreements FP7-ICT-609763 TRADR.

REFERENCES

[1] R. R. Murphy, "Human-robot interaction in rescue robotics," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 2, pp. 138–153, 2004.

[2] J. Richer and J. L. Drury, "A video game-based framework for analyzing human-robot interaction: characterizing interface design in real-time interactive multimedia applications," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on human-robot interaction*. ACM, 2006, pp. 266–273.

[3] J. A. Atherton and M. A. Goodrich, "Supporting Remote Manipulation with an Ecological Augmented Virtuality Interface," *(unknown)*, 2010.

[4] J. Y. C. Chen, E. C. Haas, M. J. S. M. Barnes, C. P. C. Applications, and R. I. T. on, "Human Performance Issues and User Interface Design for Teleoperated Robots," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 6, Jan. 2007.

[5] H. A. Yanco and J. L. Drury, "Rescuing interfaces: A multi-year study of human-robot interaction at the aaai robot rescue competition," *Autonomous Robots*, vol. 22, no. 4, pp. 333–352, 2007.

[6] C. W. Nielsen, M. A. Goodrich, and R. W. Ricks, "Ecological interfaces for improving mobile robot teleoperation," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 927–941, 2007.

[7] A. Kelly, N. Chan, H. Herman, D. Huber, R. Meyers, P. Rander, R. Warner, J. Ziglar, and E. Capstick, "Real-time photorealistic virtualized reality interface for remote mobile robot control," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 384–404, 2011.

[8] M. Micire, J. L. Drury, B. Keyes, and H. A. Yanco, "Multi-touch interaction for robot control," in *Proceedings of the 14th international conference on Intelligent user interfaces*. ACM, 2009, pp. 425–428.

[9] B. Larochelle and G. Kruijff, "Multi-view operator control unit to improve situation awareness in usar missions," in *RO-MAN, 2012 IEEE*. IEEE, 2012, pp. 1103–1108.

[10] P. Ögren, P. Svenmarck, P. Lif, M. Norberg, and N. E. Söderbäck, "Design and implementation of a new teleoperation control mode for differential drive ugvs," *Autonomous Robots*, vol. 37, no. 1, pp. 71–79, 2014.

[11] S. Muszynski, J. Stuckler, and S. Behnke, "Adjustable autonomy for mobile teleoperation of personal service robots," in *RO-MAN, 2012 IEEE*. IEEE, 2012, pp. 933–940.

[12] M. Mast, Z. Materna, M. Španěl, F. Weisshardt, G. Arbeiter, M. Burmester, P. Smrž, and B. Graf, "Semi-autonomous domestic service robots: Evaluation of a user interface for remote manipulation and navigation with focus on effects of stereoscopic display," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 183–202, 2015.

[13] A. E. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow, "Strategies for human-in-the-loop robotic grasping," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. ACM, 2012, pp. 1–8.

[14] E. You and K. Hauser, "Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input," in *Robotics: science and systems*, vol. 7, 2012, p. 354.

[15] M. Frejek and S. B. Nokleby, "A methodology for tele-operating mobile manipulators with an emphasis on operator ease of use," *Robotica*, vol. 31, no. 03, pp. 331–344, 2013.

[16] L. B. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, 1993, pp. 76–82.

[17] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager, "Vision-assisted control for manipulation using virtual fixtures," *Robotics, IEEE Transactions on*, vol. 20, no. 6, pp. 953–966, 2004.

[18] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond, "Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2939–2944.

[19] Y. Wang, F. Vina, Y. Karayiannidis, C. Smith, and P. Ogren, "Dual arm manipulation using constraint based programming," in *IFAC World Congress, Cape Town, South Africa*, 2014.

[20] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1994.