--Preprint--

An Autonomous Dynamic Camera Method for Effective Remote Teleoperation

Daniel Rakita, Bilge Mutlu, Michael Gleicher
Department of Computer Sciences, University of Wisconsin–Madison
{rakita,bilge,gleicher}@cs.wisc.edu

ABSTRACT

In this paper, we present a method that improves the ability of remote users to teleoperate a *manipulation* robot arm by continuously providing them with an effective viewpoint using a second *camera-in-hand* robot arm. The user controls the manipulation robot using *any* teleoperation interface, and the camera-in-hand robot automatically servos to provide a view of the remote environment that is estimated to best support effective manipulations. Our method avoids occlusions with the manipulation arm to improve visibility, provides context and detailed views of the environment by varying the camera-target distance, utilizes motion prediction to cover the space of the user's next manipulation actions, and actively corrects views to avoid disorienting the user as the camera moves. Through two user studies, we show that our method improves teleoperation performance over alternative methods of providing visual support for teleoperation. We discuss the implications of our findings for real-world teleoperation and for future research.

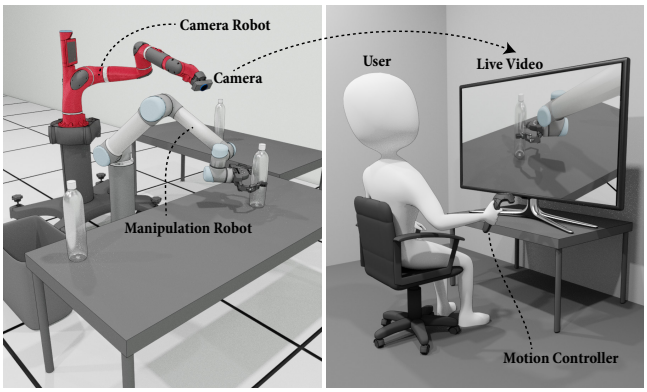
ACM Reference Format:
Daniel Rakita, Bilge Mutlu, Michael Gleicher. 2018. An Autonomous Dynamic Camera Method for Effective Remote Teleoperation. In *HRI '18: 2018 ACM/IEEE International Conference on Human-Robot Interaction, March 5–8, 2018, Chicago, IL, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3171221.3171279

1 INTRODUCTION

Robot teleoperation holds significant near-term promise through applications that extend human motor ability, allowing users to perform manipulations in remote or extreme environments that are inaccessible or unsafe for direct human operation [21]. While a substantial body of work has explored the *control* of robot systems for teleoperation [25], less work explores how to provide a *view* of the remote environment in a way that supports effective manipulations. Visual support is critical for the success of teleoperation, as the quality of the control or the interface cannot overcome an obfuscated or unclear view of the manipulation space.

Approaches exist for viewing a remote environment during teleoperation, such as using an array of static cameras placed around the workspace or attaching a camera to the robot's end effector, however these approaches have several limitations. For instance, when static cameras are used, the manipulation robot commonly blocks the user's view on one or many of the cameras at a time, requiring the user to piece together information from multiple views, potentially increasing user cognitive load and task complexity. When end-effector cameras are used, the user cannot get sufficient context, i.e., a view of how the robot is situated in the environment, making costly collisions likely. Additionally, in grasping tasks, the user's view is partially or completely blocked once the gripper is holding an object due to the position of the end-effector camera. Although these approaches may sufficiently support simple teleoperation tasks, critical real-world scenarios, such as remote home-care, tele-nursing, space exploration, or managing nuclear materials, require a robust solution that addresses these limitations.

Figure 1: We present a method that improves remote teleoperation by continuously providing users with an effective viewpoint of the remote environment. A *camera robot* smoothly servos in real-time as the user teleoperates the *manipulation robot* based on the optimized views.



In this paper, we provide an alternative: providing a camera that automatically moves to provide an effective view during manipulation. By jointly optimizing over a *camera location* and *viewing direction* in real-time, such a "moving-camera" approach can automatically adapt the view of the remote environment to the user's changing visual needs. We present a method that provides an automatic dynamic camera using a second "camera-in-hand" robot arm alongside the "manipulation" robot arm that continuously tries to provide a clear viewpoint for the user in real-time. In our method, as the user controls the manipulation robot using a teleoperation interface, the "camera-in-hand" robot servos in real-time to provide a sufficient view of the remote environment while avoiding occlusions with the manipulation robot, offering both context and detailed views of the environment, and utilizing motion prediction to highlight next user actions.

In our solution, we chose to make the camera robot's motion *autonomous*, as opposed to requiring users to control both robots, in order to reduce the cognitive load and allow user to focus on manipulation. This approach builds on prior research on teleoperation that shows that users perform better when reasoning over fewer degrees of freedom during control [18, 27]. However, while automatically moving the camera to consistently provide a sufficient view of the workspace has many benefits, viewpoint translations and rotations involved in autonomous movement may also disorient the user. Our method addresses this challenge by updating the control scheme of the manipulation robot in a way that maximizes viewpoint stability for the user as the camera moves.

We implemented our automatic-camera method into a prototype system shown in Figure 1. In our implementation, the user controls a Universal Robots UR5 robot arm,[1] while a Rethink Robotics Sawyer robot arm[2] with a camera in hand continuously optimizes a viewpoint for the user in real-time. The camera streams a video feed of the manipulation environment that is shown to the user on a screen. The user controls the UR5 robot arm using the *mimicry-control* method described by Rakita et al. [22], using an HTC Vive Motion Controller[3] as the motion capture device.

The specific contributions of our work include (1) the introduction of the idea of an *automatic dynamic camera* to improve remote teleoperation performance, (2) the set of optimization procedures used to sufficiently instantiate an automatic-camera method using a second "camera-in-hand" robot arm (§4); (3) a novel way to dynamically update the representation of the user's control inputs to account for the moving camera (§4.3); and (4) a demonstration of the effectiveness of our automatic-camera method and control algorithm against baseline methods through two user studies (§5).

2 RELATED WORK

Our automatic dynamic camera method to provide visual assistance in teleoperation draws from prior work, especially in active vision, visual servoing, and teleoperation. We also build on methods to automatically place and move virtual cameras in a scene.

Active Vision—Active vision involves methods that systematically determine the pose of a visual sensor on a robot platform (see Chen et al. [7] for a detailed review). Work in this area reasons about posing cameras for applications such as object search [23], object modeling [3, 8], robot grasp planning [20], object tracking [4], and surveillance [24]. Our work shares similar goals with surveillance applications, such as maximizing visual coverage and avoiding occlusions [1, 24]. We additionally consider work in laparoscopic robotic surgery, which allows surgeons to control a flexible robot camera to obtain a sufficient view of the procedure area [17]. Our work attempts to solve similar problems, such as streaming back a sufficient view of a workspace for a user. However, because manually controlling the camera along with the manipulation tool may require an expert user, such as a surgeon, we choose to pursue fully automated movement of the camera to support novice users.

Visual Servoing—Visual servoing is a robot-control paradigm in which a robot moves based on visual feedback (see the work by Corke [11] for a full introduction). While our work would not be considered visual servoing because the camera robot moves based on user input and the manipulation robot's geometry as opposed to vision, our work shares some parallels with this area. Our work draws from the design of *eye-in-hand* visual servoing systems, as the camera robot provides a view using a camera mounted on its end effector. Wilson et al. [26] outlines a control algorithm to grasp an object using a 6-DOF manipulator by estimating the position and orientation of the object using visual feature points. Our work does not recognize visual feature points to drive the camera robot, but we similarly use geometric anchor points to constrain the motion of the robot and camera to achieve high-level goals.

Animation & Graphics—Considering sufficient viewpoints in a virtual scene would be useful in many computer-graphics and animation applications (for an overview of camera control techniques used in computer graphics, see the work by Christie et al. [10]). Gleicher and Witkin [14] pioneered a technique called "through-the-lens" camera control, which controls a virtual camera based on what is seen through the camera rather than extrinsic camera parameters. Our control method was inspired by the effect of this method, as the coordinate frame that user inputs are represented in update based on what is seen on-screen rather than just being represented in an absolute frame. Virtual camera methods have been developed to avoid visual occlusions with objects in the scene [9]. Our visual-occlusion-avoidance method differs from such approaches as these methods can leverage full geometric understanding of the environment and can produce free-flying camera motions in the virtual scene. Galvane [13] reviews many approaches that automatically move a camera around in a virtual scene for applications such as cinematic replays and tracking people in crowd simulations. Our method draws on this work, as we dynamically move a camera in the workspace to improve the visibility of remote teleoperation.

3 OVERVIEW OF APPROACH

Our work builds on the premise that providing a clear and consistent viewpoint for the user will improve remote teleoperation and that achieving this goal will require at least minimal solutions to the five technical challenges that we outline below. We provide an overview of our solutions in §4.

Challenge 1—What position in space should serve as the camera robot's visual target?

Challenge 2—What viewing direction should the robot choose in order to clearly display its visual target?

Challenge 3—What distance should the camera maintain from its visual target? Should target view be"close-up," "wide," etc.?

Challenge 4—How should the camera robot move in real-time to pursue the visual goals decided upon in *Challenges 1–3*?

Challenge 5—How should teleoperation controls be updated on-the-fly to account for a dynamic viewpoint and to avoid camera movements that may disorient the user?

3.1 Relaxed-IK Review

Solutions to the technical challenges above require many kinematic subgoals to be pursued simultaneously. Subgoals may even be in conflict, e.g., if the robot aims to provide a view of the visual target from above, but views from above are occluded by the manipulation

[1]Universal Robots: https://www.universal-robots.com/products/ur5-robot/
[2]Rethink Robotics: http://www.rethinkrobotics.com/sawyer/
[3]HTC Vive: https://www.vive.com/us/accessory/controller/

robot. Thus, to successfully realize the automatic-camera method, the camera robot has to quickly weigh many subgoals and choose a strategy that reflects what is most important at the moment.

To accommodate all of these sub-goals in real-time, we use an optimization-based inverse kinematics solver that handles trade-offs between different objectives on the fly. At each update, the method calculates joint angles that will exhibit these desired features through a process called *inverse kinematics (IK)* (see Buss [5] for a review of common IK methods). We note that this is a *generalized* IK formulation, because we reason over kinematic goals other than just end effector position and orientation goals.

Our method utilizes the *Relaxed-IK* solver included in the mimicry-control interface described by Rakita et al. [22]. The solver utilizes a flexible optimization framework to handle IK problems that dynamically trade-off between multiple objectives.

The IK problem is formulated as a constrained optimization:

$$\Theta = \arg\min_{\Theta} \ \mathbf{f}(\Theta) \text{ s.t. } \mathbf{c}_i(\Theta) \ \geq \ \mathbf{0}, \ \ \mathbf{c}_e(\Theta) \ = \ \mathbf{0} \qquad l_i \leq \Theta_i \leq u_i, \forall i \tag{1}$$

where Θ is the $n-$vector of robot joint values (n is the robot degrees of freedom); $\mathbf{c}_i(\Theta)$ is a set of inequality constraints; $\mathbf{c}_e(\Theta)$ is a set of equality constraints; l_i and u_i values define the upper and lower bounds for the robot's joints; and $\mathbf{f}$ is a scalar objective function.

3.2 Overview of Optimization Components

To express motion qualities for the robot camera, we split our optimization into two components: (1) A *joint space solver* that reasons about *how the robot's joint states should update*; and (2) a *camera location goal solver* that focuses on *optimizing a camera location* in the remote environment. The two solvers connect with an *interface term* in the *joint space solver* optimization, which encourages the camera robot's end effector to pursue the camera location goal.

We treat finding a camera location goal as a separate optimization for two reasons: (1) This problem is more naturally solved in $\mathbb{R}^3$. In contrast, including these goals as terms in the joint space optimization would require unnecessary forward kinematics model solutions at each iteration just to translate back to Euclidean space; and (2) It is possible that our dynamic camera teleoperation approach could work by moving the camera using something other than an articulated robot arm, *e.g.,* using a drone or parallel robot. Thus, having a modular piece that just optimizes a camera location and viewing direction could generalize across platforms.

4 TECHNICAL DETAILS

In this section, we outline the optimization techniques used to solve the technical challenges outlined in §3. Notation used throughout this section is presented and explained in Figure 2.

4.1 Objective Function Structure

At each system update, our method minimizes an objective function, subject to a set of constraints and bounds, to solve for the camera robot's new joint angles, as illustrated in Equation 1. Thus, the objective function must encode desired *motion qualities* while the constraints and bounds must dictate *motion requirements*.

To express our desired motion qualities, we use a weighted sum objective function, consisting of the components introduced in §3.2:

$$\overbrace{\arg\min_{\Theta_c(t)}[\sum_{i=1}^{k} w_i f_i(\Theta_c(t), \Theta_m(t), \Omega_i)] + \underbrace{w_g f_g(\Theta_c(t), \Theta_m(t), \Omega_i, \mathbf{g}^*)}_{Interface\ Term}}^{Joint\ Space\ Solver},$$
$$\underbrace{\mathbf{g}^* = \arg\min_{\mathbf{g}} \sum_{j=1}^{m} w_j \ f_j(\mathbf{g}, \Theta_c(t), \Theta_m(t), \Omega_j)}_{Camera\ Location\ Goal\ Solver} \tag{2}$$

Here, w represents *weight priors* that remain static through runtime and allow a domain expert to incorporate prior knowledge to express the relative importance of each term. The f functions assign a value to the current inputs, *e.g.*, robot configurations or a camera location, in order to encode a single sub-goal. In the optimization, lower values returned by the objective terms should correspond to inputs that better exhibit the desired camera motion qualities. The Ω denotes a set of model parameters used to construct the objective term loss function, covered in depth below.

While a weighted-sum objective function affords expressiveness by encoding each motion goal as a single term in the sum, parameter tuning of the weights can become unwieldy, often leading to unstable or divergent behavior if care is not taken. Parameter tuning would be particularly troublesome in our automatic camera optimization, as we will be dealing with many objectives that may be in conflict at any given time. Ideally, the term weights would correspond to easily explainable behavior, such as a term with weight of two being twice as important as a term with weight of one in the optimization. This behavior is not observed using standard loss functions, such as quadratic, because optimized terms can be over different units at vastly different scales (such as joint-space velocities compared to Euclidean distances in operational space).

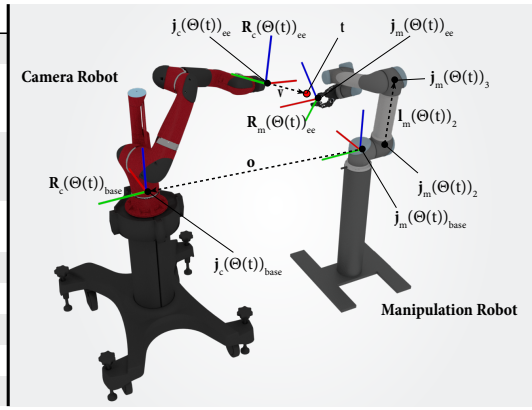
To facilitate combining objectives, we normalize each term using a parametric normalization function that is designed to scale each function to a uniform range, place a narrow "notch" around the goal values, a more gradual falloff away from the notch in order to better integrate with other objectives, and a consistent gradient that points towards the goal. We implement this normalization function as a Gaussian surrounded by a more gradual polynomial:

$$f(\Theta_c(t), \Theta_m(t), \Omega_i) = (-1)^n exp(\frac{-(\chi(\Theta_c(t), \Theta_m(t)) - s)^2}{2c^2}) + r * (\chi(\Theta_c(t), \Theta_m(t)) - s)^4 \tag{3}$$

Here, the scalar values n, s, c, r form the set of model parameters Ω. Together, they shape the loss function to express the needs of a certain term. Here, $n \in \{0, 1\}$, and dictates whether the Gaussian is positive or negative. If the Gaussian region is negative, this is an area of high "reward", while if the region is positive, the optimization will push away from this region of high "cost". The value s shifts the function horizontally, and c adjusts the spread of the Gaussian region. The r value adjusts the transition between the polynomial and Gaussian regions, higher values showing a steeper funneling into the Gaussian region, with lower values flattening out the boundaries outside of the Gaussian. The scalar function $\chi(\Theta_c(t), \Theta_m(t))$ assigns a numerical value to the camera robot and manipulation robot configurations at the current time that will serve as the variable input to the loss function.

Figure 2: Table on the left: the notation used throughout our automatic camera work. Figure on the right: the setup of the camera and manipulation robots annotated with the notation.

Notation	Explanation
$\Theta_{\{c,m\}}(t)$	Denotes the joint configuration for the camera robot (c) or manipulation robot (m).
$\mathbf{j}_{\{c,m\}}(\Theta(t))_i$	Denotes the *3D position* of the camera robot (c) or manipulation robot's (m) ith joint given joint angle values Θ at time t. The i subscript can read "ee" or "base" to denote the end effector or base joint, respectively.
$\mathbf{R}_{\{c,m\}}(\Theta(t))_i$	Denotes the *local rotation matrix* at the camera robot (c) or manipulation robot's (m) ith joint given joint angle values Θ at time t. The i subscript can read "ee" or "base" to denote the end effector or base joint, respectively.
$\mathbf{l}_{\{c,m\}}(\Theta(t))_i$	Denotes the *link vector* after the ith joint on the camera robot (c) or manipulation robot (m) given joint angle values Θ at time t $[\mathbf{j}_{\{c,m\}}(\Theta(t))_{i+1} - \mathbf{j}_{\{c,m\}}(\Theta(t))_i]$.
$\mathbf{o}$	*Position offset* from manipulation robot to camera robot $[\mathbf{j}_c(\Theta(t))_{base} - \mathbf{j}_m(\Theta(t))_{base}]$.
$\mathbf{t}$	*Visual target position* in the environment, near manipulation robot's end effector.
d	*Distance* between camera robot's end effector and camera location goal $[\|\mathbf{t} - \mathbf{g}\|]$.
$\mathbf{v}$	*View direction* vector, that eminates from $\mathbf{j}_c(\Theta(t))_{ee}$ through the camera's focal point.
$\mathbf{g}$	*Goal Position* that the camera robot's end effector should pursue.



Unless otherwise specified in the following sections, the Ω parameter values for a term are $\{n = 1, s = 0, r = 1.0, c = 0.1\}$. These values shape the loss function to encourage values to go to zero, which is a common goal in optimization (such as when minimizing energy, velocities, or distances between points). In the following sections, we will detail what $\chi(\Theta_c(t), \Theta_m(t))$ and Ω should be for each term to solve the technical challenges outlined in §3.

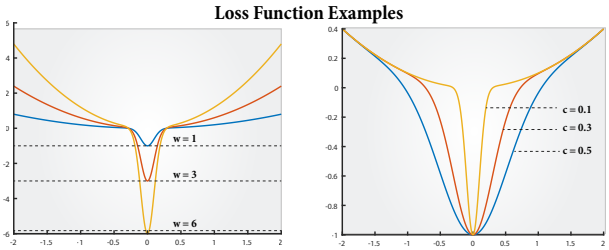
4.2 Technical Solutions

In this section, we describe the optimization used to calculate a new joint configuration for the camera robot at each system update. Each solution corresponds to a technical challenge listed in §3. We list the weight w of each objective term within the equations.

Solution to Technical Challenge 1— At each system update, our method must decide what visual-target position in space on which to focus. We make the simple assumption that, because the user interacts with the remote environment using the manipulation robot's end effector, the salient point in space should be near the manipulation robot's end-effector position. We also add in simple motion inference such that the visual target point is where the end effector will be in the near future in order to lead the user's view toward where they are moving the robot rather than reacting to where the end effector is at the current update.

In the form shown in Equation 3, we have:

Figure 3: Examples of the loss function used in our weighted-sum objective. Left: Scalar multiplication by a weight fully controls the amplitude of the reward region. Right: The value "c" controls the spread of the reward region.



$$\chi(\Theta_c(t), \Theta_m(t)) = dis(\mathbf{t}, \mathbf{v}), w = 5 \tag{4}$$

Here, dis is a function that returns the orthogonal distance between the point $\mathbf{t}$ and the line segment $\mathbf{v}$. The $\mathbf{t}$ point is the visual target, which uses a linear prediction model by extrapolating the user's input velocity direction to infer a point 0.3 s into the future.

Solution to Technical Challenge 2— Once a visual target position is determined, the method must select a viewing direction that maintains a sufficient view of the visual target. Solutions to this problem can draw on the process for deciding upon the camera robot's end-effector point, as a viewing direction is formed by connecting the camera position with the target point.

We use two criteria to select a viewing direction: (1) the view of the visual target should not be occluded by the manipulation robot; and (2) the target should typically be viewed from above the manipulation point since views from below are often not practical for standard teleoperation workspaces such as tables or the ground.

To avoid occlusions, we model the manipulation robot's links as line segments, and try to maximize the distance between the viewing direction vector and these geometric objects. These objectives are included in the *camera goal location solver*, and take the form:

$$\chi(\mathbf{g}, \Theta_c(t), \Theta_m(t)) = dis(\mathbf{v}, \mathbf{l}_m(\Theta(t))_i), \forall i \in \{1 \ldots N-1\}, w = 3 \tag{5}$$

Here, N represents camera-robot DOFs. Link N is not included, as it is near the end effector and the target and thus should not be visually avoided. The $dis(.,.)$ function in Equation 5 refers to the distance between the closest points on two separate line segments.

The occlusion avoidance terms all use loss function parameters $\Omega := \{n = 0, s = 0, r = 0.0001, c = 0.04\}$. Note that $n = 0$ here implies a positive Gaussian region, inducing a "cost" region around the origin. A c value of .04 will start to incur a noticeable cost when the distance is below 0.3 meters, and will exponentially rise up to a maximum of the term's weight value at distance zero.

We encode a preference for high camera positions because high vantage points often work well for standard teleoperation workspaces, such as a table or the ground. To do so, we use the following objective in the *camera goal location solver*:

$$\chi(\mathbf{g}, \Theta_c(t), \Theta_m(t)) = \mathbf{g}[z], w = 1 \tag{6}$$

Here, $\mathbf{g}[z]$ denotes the z component of the g vector, which corresponds to the "up" direction in our coordinate frames. To encourage z to be high, we use Ω parameter values of $n = 1, s = 0.7, r = 0.2, c = 0.2$. This ideally favors heights of around 0.7 meters, which worked well for the robot used in our prototype system, but the wide spread of the reward region caused by $c = 0.2$, allows this value to freely fluctuate between $0.5 - 0.9$ meters in height.

Solution to Technical Challenge 3— One of the main benefits of a dynamic camera is the ability to change the distance between the camera and the manipulation point on the fly, which can enable the user to switch between a *context view* to get an overall sense of the environment and a *detailed view* of points of particular interest. At each update, our method decides upon a distance between the camera and the visual target using a heuristic based on the user's input velocity to estimate a desired distance. High velocities may indicate that the user wishes to get a broader scope of the environment to see where the robot is moving. Low velocities, on the other hand, may indicate fine manipulation that would benefit from a close-up view of the manipulation point. We note that this heuristic may not be effective in all scenarios, as user preferences and task complexity may affect the relationship between input velocity and user intent. Thus, we use this input-velocity heuristic as a reasonable guess and provides users with the capability to manually update this distance using a 1D input method, such as the touchpad on the controller.

We incorporate this as an objective term in the *camera goal location* solver with the following structure:

$$\chi(\mathbf{g}, \Theta_c(t), \Theta_m(t)) = ||\mathbf{g} - \mathbf{j}_m(\Theta(t))_{ee}|| - d, w = 2 \tag{7}$$

Here, d is a goal distance that is iteratively updated based on the user's input velocity and manual inputs. In our prototype system, this could range over a distance of $0.4 - 1.4$ m.

Solution to Technical Challenge 4— In Solutions 1 through 3, we outlined visual goals that the camera should ideally achieve at each update. However, these solutions do not address *how* the camera robot should move over time to smoothly achieve these goals.

To address this challenge, we use four kinematic guidelines, each of which is encouraged by the method over time:

(1) The camera robot should move smoothly so as not to distract the user's control with shaky or discontinuous motion by minimizing velocities, accelerations, and jerks in both the robot's joint space and end-effector Cartesian space. We minimize velocities in both spaces by including objectives of the form:

$$\begin{gathered}\chi(\Theta_c(t), \Theta_m(t)) = ||\Theta_c'(t)||, w = 4 \\ \chi(\Theta_c(t), \Theta_m(t)) = ||\mathbf{j}_c'(\Theta(t))_{ee}||, w = 4\end{gathered} \tag{8}$$

Each of these terms have the same Ω parameters for the loss functions: $n = 1, s = 0, r = 1.0, c = 0.01$. The c value, which controls the spread of the Gaussian, is small here because the velocity values will be close to zero. Our method also limits accelerations and jerks using constraints that set a maximum value of $0.1\ m/s^2$ and $0.1\ m/s^3$for acceleration and jerk magnitudes, respectively.

(2) The camera robot should avoid collisions with itself and with the manipulation robot that can be unsafe and costly in scenarios such as space exploration and nuclear power management. We set hard constraints in the optimization to avoid such configurations by modeling the robot links as vectors between consecutive joint points and enforcing a minimum distance between these links.

(3) The camera robot's end effector should not rotate along its local "roll" axis so that the user is not disoriented by camera rolls. We add in an objective term that keeps the local "left" axis ("y" in our system) of the camera robot's end effector orthogonal to an "up" vector in the environment. This objective takes the form:

$$\chi(\Theta_c(t), \Theta_m(t)) = \mathbf{R}_c(\Theta(t))_{ee}[y] \cdot [0, 0, 1]^T, w = 3 \tag{9}$$

(4) The camera robot's end effector should pursue the camera goal location, determined by the *camera goal location* solver at each update. This objective contributes to the *interface term* in Equation 2, and it takes the form:

$$\chi_g(\Theta_c(t), \Theta_m(t), \mathbf{g}^*) = ||\mathbf{g}^* - \mathbf{j}_c(\Theta(t))_{ee}||, w_g = 4 \tag{10}$$

4.3 Controls with Dynamic Viewpoint

In §3, we introduce the technical challenge of how the manipulation robot controls should update as the camera dynamically moves so as not to disorient the user. Our solution is to rotate the coordinate frame that controls are represented in along with the camera's coordinate frame. Using this technique, translations and rotations can be made with respect to the user's relative view of the screen. This mitigates disorientation challenges because users simply command the robot based on their relative view of the robot on-screen at the moment rather than having to keep track of how the camera is moving with respect to a static, absolute control frame.

Position Update— Positions are updated based on the following rule: $\mathbf{p}(t) = \mathbf{p}(t-1) + \mathbf{R}_c(\Theta(t))_{ee}(\mathbf{h}'(t))$. Here, $\mathbf{h}'(t)$ is the user's velocity input at time t. Because the camera is aligned with the camera robot's end effector, rotating the user's velocity inputs by this rotation matrix transforms the inputs into the screen frame.

Rotation Update— Rotation updates follow a similar rule as position updates, but adapted to quaternions. We take raw orientation inputs from the input device as quaternions, labeled $\mathbf{q}_r(t)$, calculate a rotation velocity based on observed inputs, update this velocity based on the user's current view of the environment on-screen, then update the quaternion goal $\mathbf{q}(t)$ based on the adjusted velocity.

We first define an operator over quaternions that can effectively rotate quaternions along with the camera viewpoint: $\beta : \mathbf{q} \times \mathbf{R} \rightarrow \mathbf{q}$. Here, $\mathbf{q}$ implies a normalized rotation quaternion, $\mathbf{q} \in \mathbb{S}^3$, and $\mathbf{R}$ implies a 3×3 rotation matrix, $\mathbf{R} \in SO(3)$. Recall that a quaternion $\mathbf{q} = (iq_x + jq_y + kq_z + q_w) = (sin(\theta)\ (q_x, q_y, q_z), cos(\theta))$ rotates a point 2θ radians about the axis (q_x, q_y, q_z). This definition of quaternions *implies* a coordinate frame that this rotation takes place in. Our β operator *rotates the representation frame* that the quaternion is represented in by rotating the axis (q_x, q_y, q_z). Thus, our β operator is as follows:

$$\beta(\mathbf{q}, \mathbf{R}) = (sin(\theta)\ \mathbf{R} \cdot (q_x, q_y, q_z)^T, cos(\theta)) \tag{11}$$

Note that the quaternion does not need to be renormalized using this definition, because the rigid linear transformation induced by $\mathbf{R}$ preserves scale, leaving the q_w quaternion component untouched. Using this definition, we can formalize our update rule for rotations:

$$\begin{gathered}\mathbf{q}_{v1}(t) = \beta(\mathbf{q}_r(t-1), \mathbf{R}(\mathbf{q}_r(t))) \\ \mathbf{q}_v(t) = \beta(\ disp(\mathbf{q}_{v1}(t), \mathbf{q}_r(t)), \mathbf{R}_c(\Theta(t))_{ee}) \\ \mathbf{q}(t) = \mathbf{q}_v(t) * \mathbf{q}(t-1)\end{gathered} \tag{12}$$

Here, the *disp* is the standard displacement operator for quaternions: $disp(\mathbf{q}_1, \mathbf{q}_2) = log(\mathbf{q}_1^{-1} * \mathbf{q}_2)$ [19]. This function returns a

quaternion that encodes the rotation from $\mathbf{q}_1$ to $\mathbf{q}_2$. The function $\mathbf{R}(\mathbf{q})$ converts a quaternion $\mathbf{q}$ to a rotation matrix.

Clutching— A key aspect of our remote teleoperation interface is the ability for users to disengage control and reset to a neutral position using *clutching*, allowing users to "scroll" around the remote environment using incremental motions through pausing and unpausing, rather than constant control. Clutching is accommodated in our method by simply setting $\mathbf{p}(t)$ and $\mathbf{q}(t)$ to the last values seen right before clutching was engaged. When the user re-engages control, positions and orientations smoothly pick up where the user left off as updates are based on instantaneous parameters.

5 USER STUDIES

We carried out two user studies to demonstrate the effectiveness of our automatic dynamic camera methods for remote teleoperation. The two studies followed the same high-level designs. In this section, we describe the shared elements of both studies, leaving the specific elements of each study for subsequent sections.

5.1 Implementation of the Prototype System

We realized our automatic dynamic camera method in a system, described below, designed to provide sufficient performance and safety in order to demonstrate its benefits in a user study.

Teleoperation Interface—In our system, we used the *mimicry-control interface*, presented by Rakita et al. [22], to control the manipulation robot for remote teleoperation. This method was shown to be effective for novice users to control a robot arm using full 6-DOF Cartesian control compared to other interfaces, making it an appropriate option for our application. We utilized an HTC Vive motion controller as the input device to capture user input at 80 Hz. We adapted the mimicry-control method by interpreting the user hand motions as translation and rotation velocity inputs, facilitating control-frame adjustments and clutching for remote teleoperation.

Robots—Our system used two robot arms: a 6-DOF UR5 robot as the manipulation robot and a 7-DOF Sawyer robot as the camera robot. While we took advantage of the redundant joint on the camera robot by implicitly regularizing in the optimization, our method would be applicable to robots with varying DOFs.

System Architecture—Our system was set up as a distributed system over numerous computers that communicated using the Robot Operating System (ROS). The Vive motion capture device sent transformation information to the ROS environment through a dedicated Windows computer via UDP messages. A separate computer ran two instances of the *Relaxed-IK* solver, one to drive the manipulation robot based on user input and one to update the joint states of the camera robot. Our system used standard Logitech 930e webcams, which streamed high-definition video over USB that was then displayed on a large-screen monitor.

5.2 Overview of User Studies

Study Setup. In the physical setup for our studies, shown in Figure 1, the camera robot was placed next to the manipulation robot with a camera secured to its end effector. The camera streamed a live video feed to a monitor that provided a real-time view of the workspace. We placed physical dividers between the participants and robot workspace to simulate a remote teleoperation setting in which the user can only make adjustments based on what they see on screen. The experimenter was seated at a desk next to the participants.

Study Tasks. To ensure the generalizability of our findings to wide range of remote teleoperation tasks, we developed three tasks that followed a home-care scenario inspired by prior work in teleoperation [22]. Participants were asked to log in to a remote-teleoperation system to care for a friend or family member by disposing trash to clear off a table, putting away unused objects, and organizing pills. Both studies utilized the same three tasks, which were ordered such that task difficulty increased through the experiment.

Bottle Recycling—In the first task, participants picked up three plastic bottles and dropped them in a recycling bin. The bottles were spaced around two work tables, and the recycling bin was placed behind the robot with respect to its starting configuration. This task was easy in terms of manipulation but difficult in terms of needing to track broad motions around the workspace.

Toy Cleanup—Our second task involved participants picking up two LEGO Quatro towers and dropping them in a toy bin. In contrast to the bottle-recycling task, this task involved a more complex manipulation as the squared-off edges of the pieces required more orientation precision upon grasping, but the workspace was not as broad, as the toy bin was on the table near the objects.

Pill Organization—In our third task, three pill bottles were presented on the main work table, each with a small pill inside. The participants picked up the pill bottles one at a time, pouring the pill into one of three cups on the second table next to the robot. This task was designed to be the most difficult for manipulation and visibility; the pill bottles were difficult to grasp due to their small size, and pouring the pills required rotational dexterity and visibility around the end effector and the cups.

Study Procedures. In both studies, a male experimenter obtained informed consent and provided participants with detail on the study goals and tasks. Participants were then presented with an training video that introduced them to the robot-control approach and the motion controller. After watching the video, the experimenter guided them through an interactive training session in which participants practiced controlling the manipulation robot by picking up and moving around a water bottle while standing directly behind it until proficiency was achieved. Participants then performed the bottle task again with one eye closed or covered, which was intended to prime the participants' expectations about depth-perception challenges that would arise when using a screen-based interface.

After training, participants used a remote-teleoperation setup involving a screen-based interface and a divider that was set up between the participant and the robots such that they could only see the environment through the views on the screen. In the remainder of the study, participants (1) received training on a particular remote teleoperation condition, (2) performed the three tasks outlined in §5.2 using that condition, and (3) filled out a questionnaire pertaining to that condition. This process repeated until all conditions were completed with a short break in between while the experimenter reset the robot to the same initial configuration to standardize the starting point and set up the workspace for the new task. Once all conditions were completed, participants filled out a demographic survey and received compensation.

Measures. To assess participant performance in both studies, we measured *binary success* over the 16 subtasks involved in the three tasks: six in the *bottles task*: [*pick up bottle*, *recycle bottle*] × 3, four in the *legos* task: [*pick up lego tower*, *drop lego tower in bin*] × 2, and six in the *pills* task: [*pick up pill bottle*, *pour pill into cup*] ×3. For each task (*bottles*, *legos*, and *pills*), the participants had a maximum time of five minutes to complete as many subtasks as they could.

To measure participant perceptions of the overall control experience, we administered a questionnaire based on prior research on measuring user preferences and teamwork with a robot [12, 15], including two scales on *ease-of-use* and *fluency*, using a seven-point rating scale. Ease of use was measured using items "The control method made it easy to accomplish the task," "I felt confident controlling the robot," and "I could accurately control the robot" (Cronbach's α = 0.95 and 0.93 in Study 1 and 2, respectively). Fluency was measured using items "The robot and I worked fluently together as a team," and "The robot contributed to the effectiveness of our team" (Cronbach's α = 0.91 and 0.84 in Study 1 and 2, respectively).

5.3 Study 1: Assessing *Control* Methods

Our first study assessed the performance of our adaptive control algorithm, presented in §4.3, intended to mitigate feelings of spatial disorientation due to camera movement. Specifically, this study aimed to determine whether or not this feature of our automatic dynamic camera method contributed to teleoperation performance.

Hypotheses. Study 1 helped us test the following two hypotheses:

H1: Our adaptive control algorithm will lead to improved teleoperation performance over a standard representation of controls in a static, absolute frame. We expect rotating the control frame along with the moving camera to mitigate the participants' feelings of spatial disorientation while controls in an absolute frame, which requires keeping track of how the robot's base frame is oriented as the camera moves, to introduce additional cognitive load.

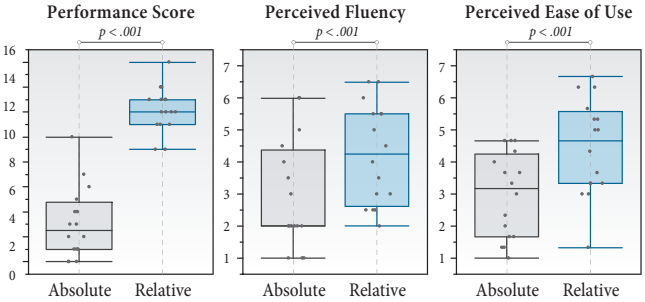
H2: Our adaptive control algorithm will be perceived as being easier to use and to provide a greater sense of fluency with the robot over standard controls in an absolute frame, because we expect our algorithm to also offer a more positive user experience.

Experimental Design. To test our hypotheses, we designed a 2 × 1 within-participants experiment in which participants completed the tasks outlined in §5.2 using two control paradigms (dynamic view frame vs. absolute frame), in a counterbalanced order.

Control Methods—Study 1 compared two control methods: *dynamic view frame* and *absolute frame*. In the dynamic-view-frame condition, participants controlled the manipulation robot, and the control frame in which inputs were represented adapted to what appeared on the screen by rotating the control frame along with the camera robot's motion, as explained in §4.3. In the absolute-frame condition, the controls were represented in the static frame of the manipulation robot's base joint, and participants had to keep in mind the camera movement relative to the static frame to correctly represent positions and rotations. This method represented standard means for displaying teleoperation input in static viewpoints, helping us to determine the benefits of a moving camera.

Participants. We recruited 16 participants for Study 1 (6 male, 10 female) from a university campus with ages 18–28 (M = 22.5, SD = 2.96). A post-hoc power analysis with $(1 - \beta) = .80$ and $\alpha = .05$ found an observed power of 0.92 (d = 3.84) with this sample size. Participants reported low familiarity with robots (M = 1.75, SD = 0.86, measured on a seven-point scale). Seven participants reported participating in prior robotics research studies. The study took 60 minutes, and each participant received $10 USD.

Figure 4: Tukey boxplots overlaid on data from objective and subjective measures for each control method.



Results. We analyzed data from all measures using repeated-measures analyses of variance (ANOVA) considering control method as the within-participants variable. Figure 4 illustrates data from our objective and subjective measures. Our analyses provided full support for both hypotheses; the adaptive-view-frame-control method significantly outperformed absolute-frame controls, $p < .001$, and was rated as being more fluent, $p = .003$, and easier to use, $p = .003$.

Discussion. In Study 1, we showed that how controls are represented in an automatic dynamic camera method for teleoperation significant affects task performance, ease of use, and sense of fluency with the robot system. Our results highlight the importance of rotating the control frame along with the view as opposed to representing controls in a fixed, absolute frame when the user's viewpoint is changing while controlling a robot.

5.4 Study 2: Assessing *Viewing* Methods

Our second study compared remote teleoperation performance using our automatic-dynamic-camera method against common remote-teleoperation-viewing alternatives.

Hypotheses. In Study 2, we formulated the following hypotheses:

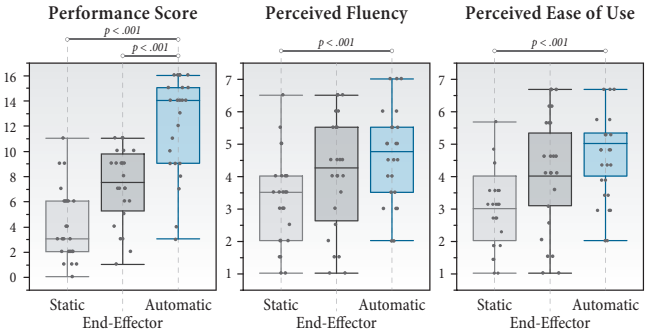
H1: The automatic dynamic camera method will outperform alternative, commonly used viewing methods in task performance, because our method adapts to the the participants' visual needs on the fly, avoiding issues such as occlusions and a narrow view.

H2: The automatic dynamic camera method will also be perceived as easier to use and will offer a greater sense of fluency with the robot over alternative viewing methods, because of the ability of our method to tailor to the visual needs of the operator.

Experimental Design. We designed a 3 × 1 within-participants study in which participants used three viewing paradigms (*automatic dynamic camera*, *array of static cameras*, *end-effector camera*), in a counterbalanced order, to perform the tasks outlined in §5.2.

Viewing Methods—In study 2, we manipulated how the users viewed the remote environment. In the *automatic dynamic camera* condition, participants used our method described throughout the

Figure 5: Tukey boxplots overlaid on data points from objective and subjective measures for each viewing method.



paper, including the adaptive-control method explained in §4.3, involving a second robot arm optimizing a viewpoint for the user.

In the *array-of-static-cameras* condition, participants viewed the remote environment through three cameras that provided a grid of three static views: a *visibility* camera that provided a view of all task-relevant objects, a *natural-view* camera placed where a user's head would be if the operator was co-located with the robot, and an *overhead* camera looking down at the entire workspace to provide un-occluded views from above. Two of these three cameras viewed the entire workspace, and two were aligned with the control frame at the base of the robot. This viewing method has been used in prior research in remote teleoperation [18].

Lastly, the *end-effector camera* condition involved securing a camera to the manipulation robot's end effector. The robot was controlled using our dynamic-view-frame controls, explained in §4.3, where controls were rotated along the manipulation robot's end-effector frame. As a common way to view environments in teleoperation and visual servoing [6], this method served as a comparison where the user manually controlled the camera.

Participants. We recruited 24 participants for Study 2 (12 male, 12 female) from a university campus with ages 18–31 (M = 22.21, SD = 3.69). A post-hoc power analysis, using $(1 - \beta) = .80$ and $\alpha = .05$, found this sample size to provide an observed power of 0.84 (d = 1.40 for automatic vs. end-effector cameras, d = 2.32 for automatic vs. static cameras). Participants reported low familiarity with robots (M = 2.26, SD = 1.38, on a seven-point scale). One participant reported participating in prior robotics research studies. The study took 90 minutes, and each participant received $15 USD.

Results. We analyzed data from all measures using one-way repeated-measures analyses of variance (ANOVA). Pairwise comparisons between *automatic-dynamic camera* and both *static cameras* and *end-effector camera* used Bonferroni correction by multiplying the p-value returned by Student's t-test by two. Data from objective and subjective measures are illustrated in Figure 5.

Our results provided full support for H1; the automatic-dynamic-camera method significantly improved teleoperation performance over the baseline methods, $p < .001$, for both comparisons. Our analyses also provided partial support for H2. The automatic-dynamic-camera method was considered easier to use than the static cameras, $p < .001$, but not the end-effector camera. Participants also rated our automatic-dynamic-camera method as being more fluent than static cameras, $p < .001$, but not the end-effector camera.

Discussion. Our findings in Study 2 showed that the automatic-dynamic-camera method outperforms baseline comparisons as a way to remotely view and teleoperate a robot. This result indicates that our method would be more effective over these alternatives when performance is critical, such as for tele-nursing, remote home care, space exploration, or nuclear-material handling.

We speculate that our method was seen as easier to use compared to the end-effector camera because both conditions used the same dynamic-view control algorithm. Also, the end-effector camera mitigated depth perception issues when grasping objects by providing an easy strategy: approach the object until the robot gripper surrounds the object. While the narrow view and occlusions made performing the task segments after the grasp difficult, we the easy grasping strategy might have provided users with a sense of accomplishment and feelings of ease of use and fluency in the task.

6 GENERAL DISCUSSION

In this paper, we present the idea of a *dynamic automatic camera* that can optimize a viewpoint on-the-fly for a user remotely teleoperating a robot arm. Our method featured a "camera-in-hand" robot that servoed in real-time to consistently provide a sufficient view for the user to perform manipulations. We instantiated our camera method in a prototype system that demonstrated the benefits of our method through two user studies. Study 1 showed that rotating the frame in which controls are represented along with the camera is an integral feature of a dynamic camera approach. In Study 2, we showed that our method significantly outperformed common alternatives of viewing a remote environment for teleoperation.

Limitations—The limitations of our current implementation and testing suggest many extensions. First, viewing a remote environment on-screen presents depth-perception challenges, because the user cannot rely on stereo-vision depth cues, which makes inferring the relative location of objects in the environment difficult. While our implementation provides some depth information through parallax when the camera moves toward or away from the target, the extent to which this effect helped in performance is unknown. We plan to explore strategies to improve depth perception in the future.

A natural extension of our method is to move the camera in a synthetic view of the environment created using real-time 3D reconstruction approaches [2, 16]. Such an approach could visually support the teleoperator by moving the camera in an unconstrained manner as long as the reconstruction of the area of interest is sufficiently detailed. Future work could compare our automatic dynamic camera method to such synthetic view approaches.

Our method is dependent on the placement and the initial joint configuration of the camera robot. While we carefully selected robot placement and configuration in our prototype system based on experimentation, a more principled method that automatically selects such parameters to maximize coverage and manipulability of the camera robot would bolster our approach.

7 ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation under award 1208632 and the University of Wisconsin–Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation.

REFERENCES

[1] Besma R Abidi, Nash R Aragam, Yi Yao, and Mongi A Abidi. Survey and analysis of multimodal sensor planning and integration for wide area surveillance. *ACM Computing Surveys (CSUR)*, 41(1):7, 2009.

[2] Dimitrios S Alexiadis, Dimitrios Zarpalas, and Petros Daras. Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15(2):339–358, 2013.

[3] Joseph E Banta, LR Wong, Christophe Dumont, and Mongi A Abidi. A next-best-view system for autonomous 3-d object reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(5):589–598, 2000.

[4] Joao P Barreto, Luis Perdigoto, Rui Caseiro, and Helder Araujo. Active stereo tracking of n < 3 targets using line scan cameras. *IEEE Transactions on Robotics*, 26(3):442–457, 2010.

[5] Samuel R Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19):16, 2004.

[6] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[7] Shengyong Chen, Youfu Li, and Ngai Ming Kwok. Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research*, 30(11):1343–1377, 2011.

[8] SY Chen and YF Li. Vision sensor planning for 3-d model acquisition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5):894–904, 2005.

[9] Marc Christie and Jean-Marie Normand. A semantic space partitioning approach to virtual camera composition. In *Computer Graphics Forum*, volume 24, pages 247–256. Wiley Online Library, 2005.

[10] Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera control in computer graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218. Wiley Online Library, 2008.

[11] Peter I Corke. Visual control of robot manipulators-a review. *Visual servoing*, 7: 1–31, 1993.

[12] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q.*, 13(3):319–340, 1989.

[13] Quentin Galvane. *Automatic Cinematography and Editing in Virtual Environments*. PhD thesis, Grenoble Alpes, 2015.

[14] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 331–340. ACM, 1992.

[15] Guy Hoffman. Evaluating fluency in human-robot collaboration. In *International conference on human-robot interaction (HRI), workshop on human robot collaboration*, volume 381, pages 1–8, 2013.

[16] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[17] Louis R Kavoussi, Robert G Moore, John B Adams, and Alan W Partin. Comparison of robotic versus human laparoscopic camera control. *The Journal of urology*, 154 (6):2134–2136, 1995.

[18] David Kent, Carl Saldanha, and Sonia Chernova. A comparison of remote robot teleoperation interfaces for general object manipulation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 371–379. ACM, 2017.

[19] Jehee Lee. Representing rotations and orientations in geometric computing. *IEEE Computer Graphics and Applications*, 28(2):75–83, 2008.

[20] Yuichi Motai and Akio Kosaka. Hand–eye calibration applied to viewpoint selection for robotic vision. *IEEE Transactions on Industrial Electronics*, 55(10): 3731–3741, 2008.

[21] Günter Niemeyer, Carsten Preusche, and Gerd Hirzinger. Telerobotics. In *Springer Handbook of Robotics*, pages 741–757. Springer, 2008.

[22] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. A motion retargeting method for effective mimicry-based teleoperation of robot arms. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 361–370. ACM, 2017.

[23] Ksenia Shubina and John K Tsotsos. Visual search for an object in a 3d environment using a mobile robot. *Computer Vision and Image Understanding*, 114(5): 535–547, 2010.

[24] Garimella SVS Sivaram, Mohan S Kankanhalli, and KR Ramakrishnan. Design of multimedia surveillance systems. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 5(3):23, 2009.

[25] Jean Vertut. *Teleoperation and robotics: applications and technology*, volume 3. Springer Science & Business Media, 2013.

[26] William J Wilson, CC Williams Hulls, and Graham S Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, 1996.

[27] Erkang You and Kris Hauser. Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input. In *Robotics: science and systems*, volume 7, page 354, 2012.