MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Pose-Based Visual Servocontrol with Keypoint Tracking

Deng, Yunfu; Chatterjee, Sreejani; Taylor, Nichols; Nikovski, Daniel N.

TR2026-082 June 18, 2026

Abstract

We propose a method for visual servocontrol of rigid bodies in the eye-to-hand setting, where the controlled body's movement is observed by a stationary RGB-D camera. The method is based on establishing and maintaining correspondences between keypoints on the body's surface across multiple images of the body in various configurations, and leverages recent advances in keypoint tracking and matching methods based on deep learning. The proposed method is pose-based, using the 3D positions of the keypoints extracted by the depth camera to estimate the relative pose of the body with respect to a reference pose. Furthermore, the method identifies the true configuration space of the controlled body by performing principal component analysis on the computed relative poses over a training sequence and decouples the control loop along each identified configuration variable.

The 15th Asian Control Conference (ASCC) 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Pose-Based Visual Servocontrol with Keypoint Tracking

Yunfu Deng, Sreejani Chatterjee, Nichols Taylor and D. Nikovski
Mitsubishi Electric Research Labs
Cambridge, Massachusetts, 02139
email: {ydeng,schatterjee,ntaylor,nikovski}@merl.com

***Abstract*—We propose a method for visual servocontrol of rigid bodies in the eye-to-hand setting, where the controlled body's movement is observed by a stationary RGB-D camera. The method is based on establishing and maintaining correspondences between keypoints on the body's surface across multiple images of the body in various configurations, and leverages recent advances in keypoint tracking and matching methods based on deep learning. The proposed method is pose-based, using the 3D positions of the keypoints extracted by the depth camera to estimate the relative pose of the body with respect to a reference pose. Furthermore, the method identifies the true configuration space of the controlled body by performing principal component analysis on the computed relative poses over a training sequence and decouples the control loop along each identified configuration variable.**

***Index Terms*—Learning control, visual servocontrol, deep neural networks, system identification**

I. Introduction

The field of visual servocontrol (VS) is concerned with controlling various mechanical systems, for example robotic manipulators or mobile robots, based on observations of the mechanism obtained by means of one or more cameras [1]. We consider the problem of controlling a robot observed by a stationary camera and actuated in velocity control mode, but not equipped with precise position encoders. This is a common setting for systems such as gantry cranes, mobile robots, and lower-cost linear stages, where motor encoders are not available or, even if available, do not provide accurate position readings for the actual robot due to slippage of belts, cables, and wheels. It is economically appealing to control such mechanisms entirely from visual observations, due to the ever decreasing cost and increasing resolution of cameras, as well as the introduction of depth (RGB-D) cameras that provide 3D measurements.

VS methods employ the basic principle of servocontrol, that is, closed-loop control of an actuator so that a measured output observation reaches or follows a commanded reference. In this, the servocontroller continuously measures the output, compares it to the reference, and drives the actuator to reduce the error. However, when the output observation is an image, as in VS, forming the feedback error directly as the difference $I^* - I_k$ between the desired reference image I^* and the current image I_k at control step k is not very helpful, because such a difference between two images is not easy to relate to a suitable value $\mathbf{u}_k$ of the control signal that needs to be applied. For this reason, VS methods usually extract some derived representation of the state of the controlled system and form the feedback error in that space.

In pose-based VS (PBVS), the control loop is based on the estimated pose (position and orientation) of the camera relative to the target or scene. First, the system estimates the 3D pose of the camera. Then, this estimated pose is compared to a desired (goal) pose, which defines where the camera needs to be with respect to the scene. Finally, a control command (often in velocity or sometimes force form) is computed to reduce this pose error, driving the system toward the desired pose [1].

A big advantage of PBVS is that it essentially reduces visual servocontrol to control in the Cartesian space of the robot, where many existing methods for task-space robot control can be applied. However, the biggest challenge in PBVS is reliable and accurate pose estimation. A common approach to pose estimation involves matching image elements, such as corners, edges, or fiducial markers, to their counterparts in a 3D description of the scene. While often accurate, the big disadvantage of this approach is that such a 3D description of the scene needs to be built manually. This task typically requires CAD models of the objects and knowledge about their exact position in the scene, making it difficult, laborious, and often impossible.

In contrast to PBVS, in image-based VS (IBVS), the state representation used in control is a vector of features $\mathbf{s} = \mathbf{s}(\mathbf{m}(I))$ derived from measurements $\mathbf{m}(I)$ obtained from the image, such as the image-plane coordinates of a set of distinctive keypoints, or the size and orientation of reliably identifiable shapes in the image [1]. A velocity VS controller can be designed by relating the joint velocity of the mechanism $\mathbf{v}$ to the velocity of the feature vector: $\dot{\mathbf{s}} = \mathbf{J_s}\mathbf{v}$, where the feature Jacobian $\mathbf{J_s}$ related to $\mathbf{s}$ consists of the partial derivatives of each feature in $\mathbf{s}$ with respect to each component of the joint velocity $\mathbf{v}$. A simple VS controller can then be obtained by the control law $\mathbf{u} = -\lambda \mathbf{J_s^+ e}$, where $\mathbf{J_s^+}$ is the Moore-Penrose pseudoinverse of $\mathbf{J_s}$, λ is a control gain, and $\mathbf{e} = \mathbf{s}(\mathbf{m}(I)) - \mathbf{s}(\mathbf{m}(I^*))$ is the feedback error computed in feature space [1].

The main challenges in IBVS are thus twofold: first, the computation of the feature vector $\mathbf{s}$ and second, the evaluation of the feature Jacobian $\mathbf{J_s}$. Some IBVS methods use problem-dependent features, such as the size and orientation of known objects in the scene, but this approach requires manual feature engineering and is thus laborious and costly. A much more

general approach is to use as features the image coordinates of distinct keypoints in the scene. In this case, feature computation amounts to reliably identifying the same set of keypoints in a sequence of images – a fundamental task in many computer vision applications. This can be accomplished either by means of tracking keypoints from one frame to the next, or their re-identification between non-successive frames [2]. In both cases, this needs to be done reliably over potentially long periods of time, so that the VS operation can be completed from start to goal.

Earlier methods for keypoint re-identification detected keypoints in the image as patches that are distinctive in appearance, for example corners, computed a scale-invariant descriptor of the patches, and established correspondence between keypoints in multiple images by matching their descriptors. The SIFT descriptor [3] and its variants, such as FAST, SURF, and ORB descriptors [4], [5] can often match keypoints with high accuracy, but also typically produce a significant portion of incorrect matches. Recent advances in deep learning technology have further increased significantly the accuracy of keypoint tracking and re-identification by learning filter banks to compute feature descriptors and also leveraging the spatial correlation between features. For example, the SuperPoint algorithm [6] trains a deep neural network (DNN) to compute a set of keypoint descriptors and also predict whether a particular pixel in an image corresponds to one of these keypoints. The resulting DNN is tailored specifically to the task of matching keypoints reliably, but is otherwise task- and scene-independent. SuperGlue is an algorithm that takes the application of deep learning further and learns the optimal way to match features itself [7]. More recently, XFeat [8] introduced an efficient learned feature detector and descriptor designed for real-time applications, paired with a lightweight attention-based matcher called LighterGlue. Unlike SIFT, whose descriptors are designed to be invariant to image rotation, XFeat descriptors are not rotation-invariant, which can limit matching accuracy when the observed body undergoes large in-plane rotations. Similarly, the Transporter network [9] learns how to track explicitly a fixed number of keypoints in images, selecting them in the process. Recent state-of-the-art feature trackers include TAPIR [10] and CoTracker3 [11].

The second fundamental task of IBVS, the evaluation of the feature Jacobian $\mathbf{J_s}$, can sometimes be performed analytically. For example, the recently proposed RoboTAP method [12] reduces robotic manipulation of objects to transport of image keypoints tracked by the TAPIR tracker by means of IBVS, where the movement of the robot is restricted to four degrees of freedom (DoF), such that two of the translational DoFs x and y are parallel to the image plane of the in-hand camera, the third translational DoF z is perpendicular to the image plane, and the fourth, rotational DoF is a rotation about z. In this setting, the image Jacobian ends up being constant, up to a factor depending on the unknown depth of keypoints, and can be computed analytically. However, in the general setting, the image Jacobian is neither constant nor can be computed analytically, so finding a suitable Jacobian can be a major impediment to using IBVS.

To address these shortcomings of both traditional PBVS and IBVS schemes, we propose a novel method for PBVS that still uses keypoint re-identification and tracking, like IBVS, but does not need a 3D scene description, unlike most PBVS methods. The method is described in Section II, and an empirical verification is presented in Section III. Section IV discusses future work and concludes the paper.

II. PBVS with Keypoint Re-Identification and Tracking

The purpose of the proposed method is to perform VS of a rigid body, such as a mobile robot or crane load that moves in space due to forces applied by actuators with control input $\mathbf{u}$. The motion of the body can possibly be restricted to a subspace, such as a plane with unknown orientation. The body is observed by a stationary RGB-D camera and an image I^* of the body in its goal pose is provided. The objective of the controller is to bring the body to its desired pose, such that the current image I registered by the camera coincides with the target image I^*. A dynamical model of the robot is not given and neither is a geometric description of the scene, including the shape and appearance of the robot.

The main idea of the method is to perform PBVS, but unlike typical PBVS schemes that compute the current and target poses with respect to a world frame and then use these two poses to compute the current relative pose with respect to the target, the proposed method computes this relative pose directly from keypoint correspondences between the current and target images. Then, the method maps the computed relative pose (in Cartesian space) to the effective configuration space of the controlled mechanism, and performs feedback control in this configuration space. In order to do this, the controller needs to construct first the configuration space of the motion of the controlled body and learn how to map images to this configuration. This is done in an identification stage on a short sequence of observed images. During the deployment stage of the controller, the learned mapping to configuration space is used in visual servocontrol. The identification and deployment stages are explained below.

A. Identification of Configuration Space

The method first applies a suitable random excitation policy to the controlled body, such that a sequence I_k, $k = 1, \dots, T$, of T images of the mechanism in motion are collected by the stationary camera at sampling times $t_k = k\Delta t$, where Δt is the sampling interval. After that, a set S of N distinctive keypoints are identified in the first image I_0 and tracked by a keypoint tracker over all images, resulting in tracks of the form $[u_{ik}, v_{ik}]$ for $i = 1, \dots, N$ and $k = 1, \dots, T$, where $[u_{ik}, v_{ik}]$ are the image coordinates of point i at control step k. Using $[u_{ik}, v_{ik}]$, the depth buffer of the RGB-D camera, and the camera intrinsics, the 3D coordinates of the point are extracted in the camera's frame, resulting in 3D tracks of the form $\mathbf{p}_{ik} = [x_{ik}, y_{ik}, z_{ik}]^T$ for $i = 1, \dots, N$ and $k = 1, \dots, T$. In general, some of these points might be occluded at certain times, so the tracker is also expected to output Boolean indicators o_{ik} indicating whether point i is visible at time step k.

The second step is to estimate the relative pose $(\mathbf{R}_k, \mathbf{t}_k)$ of the body with respect to a chosen reference pose implied by a reference image I_0, where $\mathbf{R}_k$ is the relative rotation and $\mathbf{t}_k$ is the relative translation. A good choice for a reference image is the first image in the sequence $I_0 = I_1$, implying that $\mathbf{R}_1$ is the identity matrix and $\mathbf{t}_1 = 0$. In general, given a reference frame I_0 with N keypoints measurements $\mathbf{p}_{i0} = [x_{i0}, y_{i0}, z_{i0}]^T$ for $i = 1, \ldots, N$ and $k = 1, \ldots, T$ in it, and a current image frame with corresponding measurements $\mathbf{p}_{ik} = [x_{ik}, y_{ik}, z_{ik}]^T$ at time k, the relative pose between the current and reference frame can be computed by means of the Kabsch-Umeyama (KU) algorithm [13], as long as at least 3 non-collinear keypoints are used. (In general, not all N points might be visible at time k, so only those with visibility indicator $o_{ik} = True$ will be used in this computation.) To increase the robustness of computation with respect to possible incorrect correspondences, algorithms such as RANSAC can be employed.

The third step is to construct the effective configuration space $\boldsymbol{q} \in \mathbb{R}^d$ from the sequence of relative poses $(\mathbf{R}_k, \mathbf{t}_k)$ and compute the general mapping $\boldsymbol{q} = f(\mathbf{R}, \mathbf{t})$ from any relative pose to its corresponding configuration. As this mapping computes configurations from Cartesian poses, it serves as inverse kinematics for the controlled robot.

The dimensionality d of the configuration space of the body is equal to the effective degrees of freedom (DoF) of the body. It can be discovered by analyzing the principal components of the sequence of estimated relative translations $\mathbf{t}_k$ and rotations $\mathbf{R}_k$, $k = 1, \ldots, T$. For translations, this can be done by computing the singular value decomposition (SVD) of the $3 \times T$ matrix $\mathcal{T} = [\mathbf{t}_1\ \mathbf{t}_2\ \ldots\ \mathbf{t}_T]$. The number of non-zero singular values is equal to the number d_t of translational DoF between the two bodies. If the SVD is $\mathcal{T} = \mathbf{U}\Sigma\mathbf{V}$, the vector $\mathbf{q}^{(t)} = [q_1, q_2, \ldots, q_{d_t}]^T$ of translational configuration variables can then be computed as $\mathbf{q}^{(t)} = \mathbf{U}_{d_t}\mathbf{t}$, where $\mathbf{U}_{d_t}$ are the top d_t eigenvectors of $\mathbf{U}$.

Note that when the translations $\mathbf{t}_k$ are constant, but not zero, the SVD of the matrix $\mathcal{T}$ will still contain one non-zero singular value, erroneously implying a single DoF. This condition can be detected by testing for constancy of the translations before SVD is applied, or by subtracting the mean of all translations before SVD is applied. This issue will not arise when one of the images in the sequence is chosen as a reference, because in that case all $\mathbf{t}_k \approx 0$, subject to estimation errors due to imprecise tracking of the keypoints from which translations were estimated.)

Computing the effective DoF of the rotations $\mathbf{R}_k$ is not as straightforward because, unlike translations, rotations belong to the special orthogonal group $SO(3)$. It is not a linear space, but a curved manifold embedded in $\mathbb{R}^{3\times3}$, so PCA cannot be applied directly. However, if the rotation is represented in axis-angle form, $\mathbf{r} = \theta\boldsymbol{\omega}$, where θ is the rotation angle and $\boldsymbol{\omega}$ is the rotation axis, PCA can be applied.

Once the relative rotation matrices $\mathbf{R}_k$ are converted to axis-angle vectors $\mathbf{r}_k$, we construct the $3 \times T$ matrix $\mathcal{R} = [\mathbf{r}_1\ \mathbf{r}_2\ \ldots\ \mathbf{r}_T]$, and PCA is again performed on $\mathcal{R}$ by means of SVD. The number of non-zero singular values is equal to the number d_r of rotational DoF of the body's motion. The vector $\mathbf{q}^{(r)} = [q_1, q_2, \ldots, q_{d_r}]^T$ of rotational configuration variables can then be computed as $\mathbf{q}^{(r)} = \mathbf{U}_{d_r}\mathbf{r}$, where $\mathbf{U}_{d_r}$ are the top d_r eigenvectors of the SVD of $\mathcal{R}$. The total number of DoF for the body's motion then becomes $d = d_t + d_r$ and the entire configuration $\mathbf{q}$ becomes the concatenation of $\mathbf{q}^{(t)}$ and $\mathbf{q}^{(r)}$, defining a configuration space $\mathbf{Q}$. The axes of $\mathbf{Q}$ are the top eigenvectors $\mathbf{U}_{d_t}$ and $\mathbf{U}_{d_r}$.

In order to derive a controller for the body in the configuration space constructed in this manner, we also need to relate the applied control input $\mathbf{u}$ to the resulting changes in the configuration of the body $\mathbf{q}$ through a suitable model. Because the configuration space $\mathbf{Q}$ is constructed as a subspace of the relative pose $(\mathbf{R}_k, \mathbf{t}_k) \in SE(3)$ with respect to the pose implied by I_0 that is fixed in the world frame $\mathbf{W}$ (in this case, coinciding with the camera frame), $\mathbf{Q}$ is also an inertial, stationary frame. However, the axes of $\mathbf{Q}$ do not necessarily coincide with the axes of $\mathbf{W}$. (For example, even when the body is moving in the xy plane in $\mathbf{W}$, the two translational axes of the constructed $\mathbf{Q}$ might not coincide with the x and y axes of $\mathbf{W}$, and can be any two orthogonal axes lying in that plane. In fact, the PCA procedure will choose q_1 to be the coordinate along the direction that experienced the largest range of motion in the training sequence, which does not have to be x or y.) Also, although we assume the control input is applied in an inertial frame $\mathbf{F}$, this frame is not necessarily aligned with $\mathbf{W}$ or $\mathbf{Q}$. We do assume, though, that the problem is fully actuated, i.e., there is a control input for each direction of $\mathbf{Q}$.

When $\mathbf{u}$ consists of a velocity twist $[\omega, \nu]^T$ (the more typical case for VS) applied in the frame $\mathbf{F}$, under the conditions above, there will exist a constant transform ${}^{\mathbf{Q}}\mathbf{V}_{\mathbf{F}}$ that relates the applied input $\mathbf{u}$ to changes in $\mathbf{q}$: $\dot{\mathbf{q}} = {}^{\mathbf{Q}}\mathbf{V}_{\mathbf{F}}\,\mathbf{u}$; as we do not assume knowledge of it, it needs to be estimated from the training data. When $\mathbf{Q} = SE(3)$ and $\mathbf{F} = SE(3)$ (full 6 DoF), the relation is known to be [14], [15]:

$$\begin{bmatrix}\omega_Q\\ v_Q\end{bmatrix} = \mathrm{Ad}_{{}^QT_F}\begin{bmatrix}\omega\\ v\end{bmatrix}, \quad {}^QT_F = \begin{bmatrix}{}^QR_F & {}^Qt_F\\ 0 & 1\end{bmatrix}$$

$$\mathrm{Ad}_{{}^QT_F} = \begin{bmatrix}{}^QR_F & 0\\ [{}^Qt_F]_\times\,{}^QR_F & {}^QR_F\end{bmatrix}, \; [t]_\times = \begin{bmatrix}0 & -t_z & t_y\\ t_z & 0 & -t_x\\ -t_y & t_x & 0\end{bmatrix},$$

where ${}^QR_F$ and ${}^Qt_F$ are the rotation and translation components of the transform ${}^QT_F$ relating $\mathbf{F}$ to $\mathbf{Q}$, and the adjoint of that transform $\mathrm{Ad}_{{}^QT_F}$ is the needed mapping ${}^{\mathbf{Q}}\mathbf{V}_{\mathbf{F}}$ from control input to velocity in configuration space. This mapping can be estimated by least squares from pairs of control inputs $\mathbf{u}_k$ and estimated configuration velocities $\dot{\mathbf{q}}_{k+1} = (\mathbf{q}_{k+1} - \mathbf{q}_k)/\Delta t$, possibly filtered suitably [15].

In the more general case we are considering in this paper, $\mathbf{Q}$ is a subspace of $\mathbf{W}_0 = SE(3)$, the space of all relative poses with respect to the pose implied by image I_0, and the control is applied in the same subspace, but likely misaligned with the axes of $\mathbf{Q}$. If now $\mathbf{u} = [\omega', \nu']^T$ in the subspace $\mathbf{Q}$, let again $[\omega, \nu]^T$ be a corresponding vector in a full 6 DoF space $\mathbf{F} = SE(3)$, such that $\omega' = \mathbf{A}\omega$ and $\nu' = \mathbf{B}\nu$, and setting $\mathbf{A} = \mathbf{U}_{d_r}$ and $\mathbf{B} = \mathbf{U}_{d_t}$. That is, $\mathbf{F}$ would be the space from which the full spatial twist $[\omega, \nu]^T$ can be projected using the eigenvalues discovered above to produce the applied control

$\mathbf{u} = [\omega', \nu']^T$. Because the projection from $\mathbf{F}$ is lossy (from 6-dimensional to d-dimensional), we cannot fully recover the full spatial twist, but we can lift $\mathbf{u}$ using a minimum-norm lift with block pseudo-inverses: $\omega = \mathbf{A}^+\omega'$ and $\nu = \mathbf{B}^+\nu'$. The lifted twist in $\mathbf{F}$ then maps to a twist in $\mathbf{W}_0$ through the adjoint of the transform between the two spaces, as above, and is projected to configuration velocities $\dot{\mathbf{q}}$ through the same linear projections specified by $\mathbf{A}$ and $\mathbf{B}$, resulting in the $d \times d$ matrix $\mathbf{M}$ that maps the control velocity twist $\mathbf{u}$ to $\dot{\mathbf{q}}$: $\dot{\mathbf{q}} = \mathbf{M}\mathbf{u}$, with

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}^{Q}R_F\mathbf{A}^+ & 0 \\ \mathbf{B}[^{Q}t_F]_\times {}^{Q}R_F\mathbf{A}^+ & \mathbf{B}^{Q}R_F\mathbf{B}^+ \end{bmatrix}. \quad (1)$$

That is, the mapping is adjoint-like and constant, only d-dimensional, and can be estimated by least-squares again, as in [15]. The inverse of this mapping, $\mathbf{M}^{-1}$, relates desired configuration velocities $\dot{\mathbf{q}}$ to the control input $\mathbf{u}$ that would produce them, allowing decoupling of the control loop, as explained below.

Handling non-rotation-invariant descriptors: The identification procedure described above requires establishing keypoint correspondences between the reference image I_0 and each frame I_k in the sequence. When using learned feature descriptors that are not rotation-invariant, such as XFeat [8], matching accuracy degrades as the angular difference between I_0 and I_k increases. To address this, we employ a rotated reference frame strategy: we pre-compute M rotated versions $I_0^{(\theta_j)}$ for $\theta_j = j \cdot 360°/M$, $j = 0, \ldots, M-1$, by rotating I_0 around the center of the detected object by angle θ_j. Features are detected independently in each rotated reference. For each target frame I_k, the matcher is run against all M rotated references, and the reference θ^* yielding the most RANSAC inliers is selected. To preserve keypoint identity across rotated references, each detection in a rotated frame is inverse-rotated back to the coordinate system of I_0 and associated with its nearest original keypoint within a distance threshold. This approach effectively extends any non-rotation-invariant matcher to handle arbitrarily large rotations, at the cost of M matching operations per frame.

B. Visual Servocontrol in Configuration Space

When using a controller, we assume that an image I^* of the body in its target configuration is provided, as is customary in VS applications. In order to execute control in the previously constructed configuration space, the target configuration $\mathbf{q}^*$ first must be computed. To this end, correspondences between N' keypoints in the target image I^* and reference image I_0 chosen during the identification stage are established. (These are generally not the same N keypoints used in the identification stage.) Various methods based on matching of keypoint descriptors, such as SIFT, ORB, FAST, SURF, etc. can be used [2]. Again employing the KU algorithm, the (relative) goal pose $(\mathbf{R}^*, \mathbf{t}^*)$ with respect to the reference pose implied by I_0 is computed, and from there the goal configuration inferred as $\mathbf{q}^* = f(\mathbf{R}^*, \mathbf{t}^*)$ through the previously learned "inverse kinematics" $f(\cdot)$.

The control trial starts from a novel configuration, where an image of the novel scene I_1' is acquired. (In general, this image will be different from any of the images in the training sequence.) Estimation of the current configuration $\mathbf{q}_1' = f(\mathbf{R}_1', \mathbf{t}_1')$ proceeds identically to that of the goal configuration, with the pose $(\mathbf{R}_1', \mathbf{t}_1')$ estimated from N' keypoint correspondences between images I_1' and I_0. Let S' be the set of these keypoints, and let $\mathbf{P}_k$ be the $3 \times N'$ matrix of their 3D positions at control step k, with $\mathbf{P}_1$ initialized during the keypoint matching operation by means of descriptor matching. Similarly, let $\mathbf{P}_0$ contain the positions of the keypoints in S' in the reference image I_0.

The control loop then advances by executing the following four operations at each control step k:

- **Control computation:** $\mathbf{u}_k = \mathbf{M}^{-1}\mathbf{K}_p(\mathbf{q}^* - \mathbf{q}_k')$, where $\mathbf{K}_p$ is a suitable proportional gain matrix and $\mathbf{M}$ is the coupling mapping in (1). Derivative control terms can be used, too. The computed control $\mathbf{u}_k$ is applied over time Δt.
- **Keypoint tracking:** A new camera image I_{k+1}' is captured and the N' keypoints from image I_k' are tracked in the new image, producing a new measurement matrix $\mathbf{P}_{k+1}$.
- **New configuration estimation:** The new keypoint positions in $\mathbf{P}_{k+1}$ are used together with the keypoint positions in the reference frame previously computed and stored in $\mathbf{P}_0$ to estimate the new configuration $\mathbf{q}_{k+1}' = f(\mathbf{R}_{k+1}', \mathbf{t}_{k+1}')$, identically to the initial control step.
- **Convergence check:** The new configuration $\mathbf{q}_{k+1}'$ is checked against the goal configuration $\mathbf{q}^*$ for convergence within a desired tolerance; if converged, the control loop terminates, otherwise the next control step $k+1$ is executed. (For stabilization and disturbance rejection, this control loop can be executed infinitely.)

It should be noted that keypoint correspondence between the same points in different images is established in two fairly different ways in different stages of the proposed method. One way is by means of keypoint tracking, referring to determining where keypoints go in successive images in a long sequence; this is the second step in the control loop above. In contrast, keypoint re-identification refers to establishing correspondences between keypoints in two generally substantially different images that are not necessarily successive in time. This step is necessary in determining the goal pose $\mathbf{q}^*$ by comparing I^* and I_0, as well as determining the initial pose $\mathbf{q}_1'$ needed to initiate the control loop. Both of these pairs of images could be fairly different in terms of the pose of the controlled body in them, so re-identification is the way to establish correspondences.

In principle, a good re-identification method could also be used for keypoint tracking, by simply using it on consecutive images. However, high-accuracy keypoint re-identification methods such as SIFT are known to be computationally intensive, because they detect and match keypoints at several levels of an image pyramid. Moreover, typical matching algorithms generate very many candidate keypoints to track, often numbering in the thousands, and retain only a small fraction of those that can be reliably matched, further adding to the computational complexity. This makes such re-identification methods too expensive computationally for real-time control.

On the other hand, keypoint trackers offer additional advantages in terms of accuracy. To begin with, keypoint trackers can leverage the physical constraint that keypoints cannot travel very

Fig. 1: A T-shaped rigid body observed by an overhead RGB-D camera.

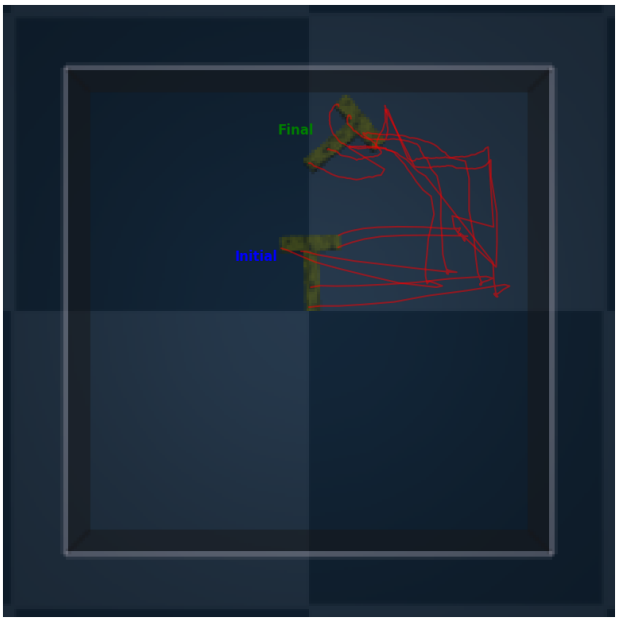

Fig. 2: The body is moved by an exploration policy to produce an identification image sequence and keypoints are tracked over the sequence by the TAPIR tracker.

far between consecutive images in order to filter out erroneous matches that a re-identification algorithm might otherwise produce. Furthermore, not only one, but several previous images in the sequence can be used to predict where a keypoint is likely to be in the next image. (Such previous images are usually not available during keypoint re-identification.) In addition, modern keypoint tracking methods based on deep learning can improve accuracy further by learning how to match keypoints optimally, as in the TAPIR keypoint tracker [10]. Other recent keypoint trackers, such as CoTracker3 [11] can also identify which keypoints tend to travel together to infer where a keypoint is likely to be in the next image based on where the other keypoints went, even when that keypoint is temporarily occluded. This increased accuracy, combined with the fast computation allowed by the inherent parallelism in executing DNNs, makes keypoint trackers much better candidates for real-time operation inside the control loop than standard re-identification methods.

III. Experimental Evaluation

To test aspects of the operation of the proposed method, we experimented with two test environments. The first consisted of a T-shape moving in a plane and observed by an overhead RGB-D camera, simulated in the MuJoCo physics engine [16] (Fig. 1). The second was a Sphero RVR+ mobile robot observed by an OAK-D Pro RGB-D camera above.

A. T-shape in MuJoCo

The T-shape body was actuated along two prismatic joints driven by linear forces and one revolute joint driven by a torque, giving it three effective DoF's (planar motion) embedded in the full 6-DoF Cartesian space. The plane of motion, as defined by the axes of the joints, are not known to the control algorithm, and the joint positions (two displacements and one angle) are not observable, either.

The initial identification sequence of $T = 120$ images is produced by executing a random exploration policy. Starting with $N = 6$ points on the T-shape, the TAPIR tracker [10] is used to track their positions in the image sequence (Fig. 2).

The identification stage of the algorithm, assuming that the reference frame is the first frame in the discovery sequence ($I_0 = I_1$) discovers that there exist two translational and one rotational DoFs and constructs a 3-dimensional configuration space. The "inverse kinematics" mapping $\mathbf{q} = f(I) = f(\mathbf{R}(I), \mathbf{t}(I))$ from any image I is computed, too, for the chosen reference image I_0.

To test the algorithm, we put the T-shape body into a new pose and capture its image I_1'. Keypoint correspondences between I_1' and the reference images I_0 are identified by means of the SIFT method [2], starting from a large number of candidate keypoints over each of the two images (Fig. 3). Because the moving body is relatively small in comparison to the background, most of these keypoints are bound to belong to the background and thus irrelevant to our controller. However, in our setting – eye-to-hand VS with a stationary camera – the background does not move. We can use this circumstance to ignore the stationary background by creating a motion mask on the moving object; this is a standard operation in computer vision algorithms [2]. By doing this, we consider for matching only those keypoints that are within the mask and thus on the moving T-shape body. Matching is done by means of a FLANN matcher of SIFT descriptors in OpenCV [17]; as this is an off-line operation, real-time computational speed is not required. To reduce the chance of erroneous matches, we apply Lowe's ratio with threshold of 0.85, i.e., we retain a correspondence only if the distance to the best match was less than 0.85 times the distance to the second-best match [3].

The image-plane coordinates of the identified keypoints in images I_1' and I_0, together with the depth at these coordinates extracted from the depth maps of the RGB-D camera, are used to compute the set of 3D matching keypoint positions in the two images. The corresponding 3D positions between the keypoints in I_1' and I_0 are then used to compute the relative pose of the controlled body in the former relative to that in the latter. As noted above, because the body has undergone a rigid-

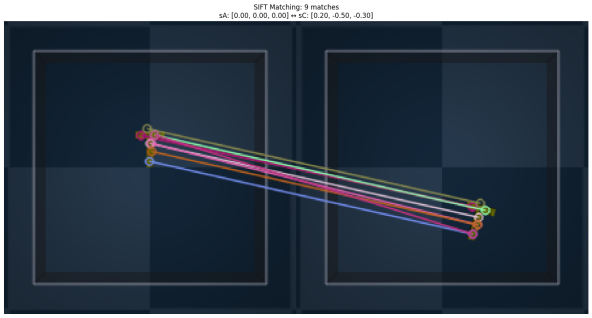

Fig. 3: Discovered keypoint correspondences between the initial configuration before PBVS starts I_1' (right) and the reference frame I_0 (left).

Fig. 4: PBVS is performed between the initial configuration (I_1', left) and goal configuration (I^*, right) while tracking the keypoints in set S'.

body transform between the two images, its rotation $\mathbf{R}_1'$ and translation $\mathbf{t}_1'$ can be recovered by means of the KU algorithm [13]. However, the SIFT matching algorithm can still return a small number of incorrect matches, as can be seen in Fig. 3, so the KU algorithm is wrapped in RANSAC for robustness. After convergence, only the inlier correspondences returned by RANSAC become members of the keypoint set S' to be used during control; essentially, these keypoints conform to a rigid body transform with respect to the reference image and are good candidates for tracking during control.

After the keypoint set for tracking S' has been determined by means of re-identification for the initial configuration I_1' (Fig. 4, left), the body is steered by means of the proposed PBVS control loop towards the goal state (Fig. 4, right). A proportional and derivative (PD) controller with proportional gain $k_p = 0.2$ and derivative gain $k_d = 0.3$ was used for all three configuration variables. The evolution of the control error in terms of the three constructed configuration variables is shown in Fig. 5. It is visible that even though slight errors in keypoint tracking do introduce some noise in control, the control is able to gradually decrease the feedback error and bring the controlled body to its desired goal configuration. For comparison, the same controller, but with ground-truth poses extracted from the physics engine instead of from keypoints is shown, too, demonstrating the effect of noise caused by tracking errors. It also clarifies that although the ranges of the configuration variables in the constructed space do not necessarily match closely those resulting from the use of actual Cartesian poses, the controller exhibits largely the same kind of regulation performance.

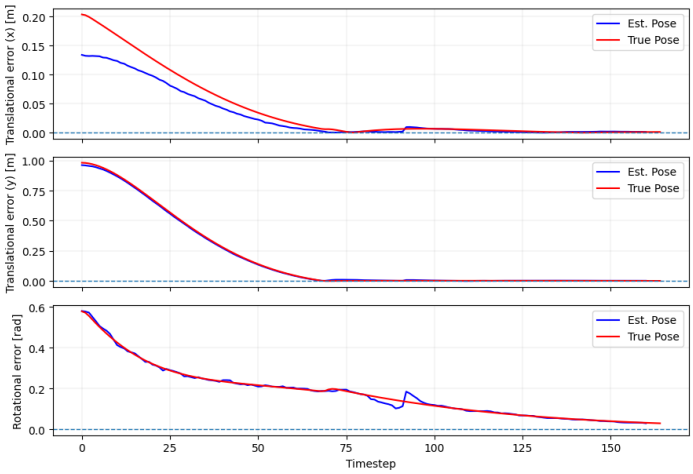

Fig. 5: Evolution of feedback error for the three configuration variables as PBVS is applied. For comparison, performance with true poses is shown, too.

B. Translational and Rotational Motion of Sphero RVR+

To verify parts of the proposed pipeline, additional experiments using a Sphero RVR+ robot ([18]) were performed. The robot was observed by an overhead OAK-D Pro camera, pointed approximately perpendicularly to the plane of motion. Unlike the simulated T-shape from the previous section, the Sphero RVR+ is nonholonomic and underactuated, similarly to most mobile robots, so full PBVS control with constant gains is generally not possible for it. (This is a consequence of Brockett's theorem which states that smooth, time-invariant stabilizing control laws do not exist for nonholonomic systems [19].) Still, the robot can be operated purely translationally or purely rotationally to collect data and verify experimentally some stages of the proposed algorithm.

To this end, the robot was commanded to move in a straight line or turn in place while being observed by the overhead RGB-D camera at 10 frames per second (fps), producing RGB images of size 640×400 pixels and aligned depth maps of the same resolution. Using the camera intrinsics provided by the camera manufacturer, full point clouds can be extracted for each recorded RGB-D image, so the x, y, z coordinate of any keypoint of interest in the camera frame can be obtained. It should be noted that the accuracy of the depth maps is expected to be on the order of 2% of the depth value, so significantly worse than the accuracy in x and y coordinates. RGB image capture, depth estimation through stereopsis, point-cloud extraction, and its delivery to a host computer takes on average 67 ± 10 ms using Luxonis's DepthAI V3 API [20], which effectively determines the fastest control rate possible.

Both linear and angular commanded velocities were constant during experiments. For translational motion, a full traversal of the visual field of the camera was performed over $T = 16$ control steps. For rotational motion, slightly more that one full turn of the robot was commanded over $T = 36$ control steps. The construction of the robot (two tracks with significant friction) minimizes slipping during motion, but slight differences in the

two tracks and their motors can possibly lead to some minimal deviations from a completely straight line when linear motion is commanded. Similarly, such differences, plus a possible tilt of the plane of motion, might lead to minimal translational motion even when purely rotational motion is commanded. Our expectation is that the dimensionality analysis stage of the algorithm will be able to capture the principal direction of motion in the space of both measured translations and rotations.

As with the T-shape, keypoint trackers as well as matchers can be used for dimensionality discovery on an initial sequence consisting of sequentially acquired image frames, but only per-frame matchers can be used from an arbitrary starting position during deployment. For this reason, we experimented with both a temporal tracker (CoTracker3 [11]) and a per-frame matcher (XFeat [8] with LighterGlue). Because XFeat descriptors are not rotation-invariant, the rotated reference frame strategy described in Section II was employed with $M = 12$ references at 30° increments. We refer to this combination as XFeat+RotRef. In all cases, the initial frame in the sequence was chosen to be the reference frame, and all relative RBTs were computed with respect to it. Moving keypoints were discovered automatically: features were seeded across the entire image and those with displacement below 10 pixels over the sequence were discarded as background, requiring no manual region-of-interest annotation. While each method produces hundreds of correspondences per frame, we visualize in Fig. 6 the full trajectories of the 6 best-matching keypoints per method, selected by lowest mean rigid-fit residual among the keypoints tracked consistently throughout the sequence. Six is the minimum number sufficient for robust RBT estimation under RANSAC while the pipeline itself operates on all available correspondences (typically $\gg 6$ per frame). Visual inspection of the trajectories shows that they capture the dominant motion of the robot, confirming that the tracker and matcher establish meaningful correspondences between the current and reference frames for relative RBT computation. Small trajectory irregularities are visible in some panels, consistent with the occasional spurious correspondence that necessitates the use of RANSAC together with KU.

The results of the SVD-based dimensionality analysis for one representative translational and one representative rotational sequence are shown in Fig. 7. For the translational sequence, the estimated 3D translation magnitude increases monotonically from 0 to approximately 75 cm, and both methods produce nearly identical trajectories (Fig. 7, top left). SVD of the $3\times T$ matrix of per-frame translations yields $\sigma_1/\sigma_2 = 13.24$ for XFeat+RotRef and 12.28 for CoTracker3, both well above the rank-1 threshold of 10 (Fig. 7, top right). For the rotational sequence, the cumulative signed rotation angle increases smoothly to approximately 400°, with both methods again in close agreement (Fig. 7, bottom left). SVD of the $3 \times T$ matrix of per-frame axis-angle vectors yields $\sigma_1/\sigma_2 = 42.09$ for XFeat+RotRef and 55.66 for CoTracker3 (Fig. 7, bottom right), confirming rank-1 rotational structure. Notably, the small residual translation observed during rotation (≤ 5 cm) is consistent with minor wheel slip and does not affect the dimensionality conclusion.

To verify that these observations hold beyond a single sequence, we aggregated the same analysis across multiple translational and rotational trials. Table I summarizes the SE(3) translation analysis across 5 translational sequences of the Sphero RVR+. We computed a performance metric called tracking quality score defined as $n_{in}/(1+r_{in})$, where n_{in} is the average number of inliers found by RANSAC over all frames and r_{in} is the average residual for these inliers, in pixels. This performance metric balances how many good correspondences a method sustains against how geometrically accurate they are; it would be maximized by as many inlier correspondences as possible, all conforming to the exact same RBT.

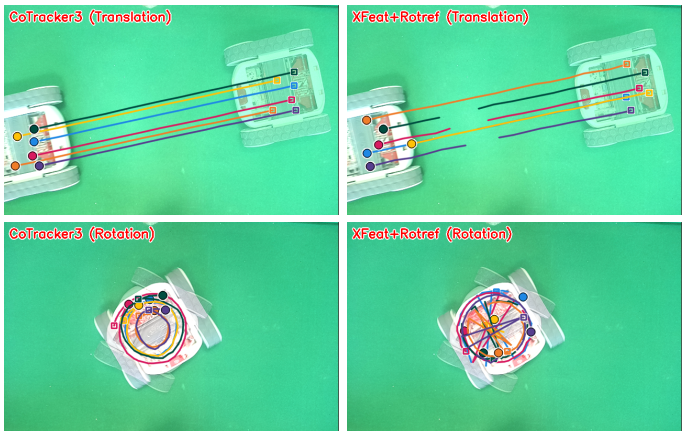

Fig. 6: Trajectories of the 6 best-matching keypoints for translational (top) and rotational (bottom) single-DoF motion of a Sphero RVR+ mobile robot, as recovered by CoTracker3 (left) and XFeat+RotRef (right). Keypoints are selected by lowest mean SE(2) rigid-fit residual among those tracked consistently across the sequence; six is the minimum number sufficient for robust RBT estimation under RANSAC, while the pipeline operates on all available correspondences per frame.

TABLE I: SE(3) translation analysis across 5 RVR+ translation sequences

Method	**Tracking Quality (range)**	**RANSAC Inliers (range)**	$\boldsymbol{\sigma_1/\sigma_2}$ **(range)**
XFeat+RotRef	99.2–105.6	307.7–332.8	9.04-14.30
CoTracker3	51.0–55.8	150.4–159.1	6.67–15.32

For each method, we report the min–max range of the tracking quality score, the mean number of RANSAC inliers per frame, and the singular-value ratios σ_1/σ_2. The latter allows clear determination that the subspace of translational motion is indeed one-dimensional. Both methods achieve rank-1 on 4 out of 5 sequences. On one sequence, both methods report σ_1/σ_2 slightly below the rank-1 threshold of 10 (9.04 for XFeat+RotRef, 6.67 for CoTracker3), consistent with a minor lateral deviation during commanded straight-line motion that was captured by both methods independently. This agreement across methods supports the view that the deviation is a property of the sequence rather than a tracking failure.

Table II summarizes the corresponding analysis across 5 rotational sequences, with a similar conclusion. These results demonstrate that the proposed pipeline, operating in fully uninformed mode with no prior knowledge of the scene or robot geometry, successfully discovers the active degree of freedom for both translational and rotational motion of a physical robot.

TABLE II: SE(3) rotation analysis across 5 RVR+ rotation sequences

Method	Tracking Quality (range)	RANSAC Inliers (range)	σ_1/σ_2 (range)
XFeat+RotRef	82.9–100.6	276–313.2	42.09–70.52
CoTracker3	36.1–48.9	116–152	3.01–55.66

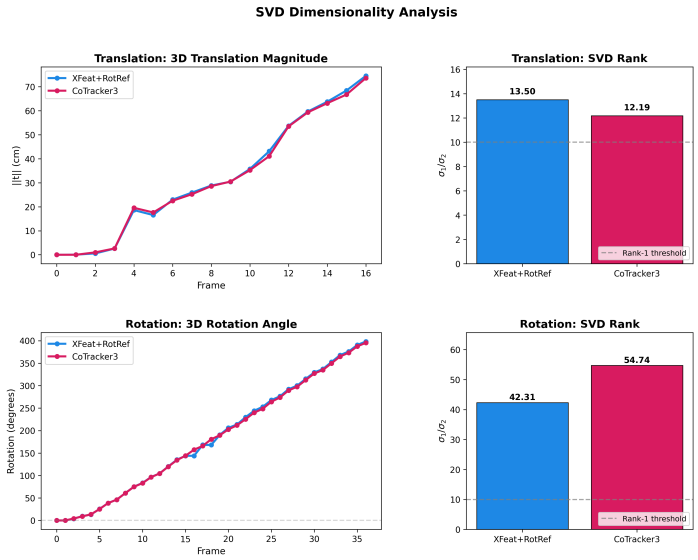

Fig. 7: SVD analysis of translational (top) and rotational (bottom) motion sequences of the Sphero RVR+. Left: estimated 3D translation magnitude and cumulative rotation angle over time, computed independently by CoTracker3 and XFeat+RotRef. Right: ratio of the first to second singular value (σ_1/σ_2) of the corresponding $3 \times T$ matrix of per-frame translations or axis-angle rotations. Both methods achieve $\sigma_1/\sigma_2 > 10$ on the active degree of freedom, confirming rank-1 structure and correct identification of one translational and one rotational DoF.

IV. Conclusion and Future Work

The paper introduced a novel method for visual servocontrol of rigid bodies, such as mobile robots, crane loads, etc. in the eye-to-hand setting, where the controlled body's movement is observed by a stationary RGB-D camera. The method is based on establishing and maintaining correspondences between keypoints on the body's surface across multiple images of the body in various configurations, and leverages recent advances in keypoint tracking methods based on deep learning. Unlike purely image-based visual servocontrol methods that employ keypoint correspondences, the proposed method is pose-based, using the 3D positions of the keypoints extracted by the depth camera to estimate the relative pose of the body with respect to a reference pose. Furthermore, the method identifies the true configuration space of the controlled body by performing principal component analysis on the computed relative poses over a training sequence and decouples the control problem along each identified configuration variable. Experiments with state-of-the-art keypoint trackers and per-frame matchers demonstrated successful identification of the configuration space of an actuated rigid body moving in an unknown three-dimensional subspace and its control by means of decoupled proportional feedback loops for each configuration variable.

The proposed method could be used for control of systems that are not equipped with accurate position encoders, for technical or economical reasons. However, the performance of the method depends critically on the ability of the keypoint tracker to establish and maintain correct correspondences between the same keypoints across time, as control progresses. For large ranges of motion, as well as cases when multiple bodies are moving in the scene, keypoint occlusion is inevitable, and dealing with it is essential for making the method applicable in practical situations. In future work, we plan to investigate the performance of the method under possible keypoint occlusion, extending it to a multi-camera set-up, as well as applying the method to articulated multi-body mechanisms such as industrial robot arms.

References

[1] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing," *Handbook of Robotics*, pp. 841–866, 2016.

[2] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[5] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[6] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[7] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[8] G. Potje, F. Cadar, A. Araujo, R. Martins, and E. R. Nascimento, "Xfeat: Accelerated features for lightweight image matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 2682–2691.

[9] T. D. Kulkarni, A. Gupta, C. Ionescu, S. Borgeaud, M. Reynolds, A. Zisserman, and V. Mnih, "Unsupervised learning of object keypoints for perception and control," *Advances in neural information processing systems*, vol. 32, 2019.

[10] C. Doersch, Y. Yang, M. Vecerik, D. Gokay, A. Gupta, Y. Aytar, J. Carreira, and A. Zisserman, "Tapir: Tracking any point with per-frame initialization and temporal refinement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 10 061–10 072.

[11] N. Karaev, I. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht, "CoTracker3: Simpler and better point tracking by pseudo-labelling real videos," *arXiv preprint arXiv:2410.11831*, 2024.

[12] M. Vecerik, C. Doersch, Y. Yang, T. Davchev, Y. Aytar, G. Zhou, R. Hadsell, L. Agapito, and J. Scholz, "Robotap: Tracking arbitrary points for few-shot visual imitation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5397–5403.

[13] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[14] K. Lynch and F. Park, *Modern Robotics*. Cambridge University Press, 2017.

[15] F. L. Markley and D. Mortari, "How to estimate attitude from vector observations," in *Astrodynamics Specialist*, 1999.

[16] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[17] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[18] Sphero, "Sphero RVR+," 2026. [Online]. Available: https://sphero.com/products/rvr?variant=42004659142701

[19] R. W. Brockett, "Asymptotic stability and feedback stabilization," *Differential geometric control theory*, vol. 27, no. 1, pp. 181–191, 1983.

[20] Luxonis, "DepthAI V3," 2026. [Online]. Available: https://docs.luxonis.com/software-v3/depthai/