Visual Inverse Kinematics: Finding Feasible Robot Poses under Kinematic and Vision Constraints

Liangting Wu and Roberto Tron

***Abstract*— We introduce Visual Inverse Kinematics (VIK), which finds kinematically feasible joint configurations that satisfy vision-based constraints, bridging the gap between inverse kinematics (IK) and visual servoing (VS). Unlike IK, no explicit end-effector pose is given, and unlike VS, exact image measurements may not be available.**

In this work, we develop a formulation of the VIK problem with a field of view (FoV) constraint, enforcing the visibility of an object from a camera on the robot. Our proposed solution introduces a virtual kinematic chain that connects the physical robot and the object, transforming the FoV constraint into a joint angle kinematic constraint. Along the way, we introduce multiple vision-based cost functions to fulfill different objectives. We solve this formulation of the VIK problem using a method that involves a semidefinite program (SDP) constraint followed by a rank minimization algorithm. The performance of this method for solving the VIK problem is validated through simulations.

I. Introduction

In robotics, the Inverse Kinematics (IK) problem is the problem of finding the values of joint configurations given a desired end-effector pose. Depending on the robot structure, there may exist an uncertain number of solutions. Solving the IK problem has been extensively researched over decades, producing numerous approaches classified as analytical [1], [2] and numerical solutions, e.g. [3]–[5]. See [6] for a review. This paper investigates a variant of the IK problem, which can be introduced using the following scenario.

Problem motivation. Imagine a robot manipulator that needs to inspect an object using a camera attached to its end-effector. For example, a manipulator makes inspections of a product as part of a manufacturing procedure. Another example is an underwater vehicle that relies on close-up vision signals from a camera mounted on its manipulator to perform operations in a low-visibility environment. Assuming the object position is known (e.g. via SLAM), our goal is to find a joint configuration that maintains the object in the field of view (FoV) while achieving some objectives, such as keeping the camera upright or viewing the object from a preferred angle. We refer to such a problem as the Visual Inverse Kinematics (VIK) problem, which is different from the Inverse Kinematics problem because the end-effector pose is not explicitly provided; instead, the feasible set is determined by the intersection between the set of the kinematic constraints of the IK problem and the set of configurations that make the object visible to the camera.

The authors are with the Department of Mechanical Engineering, Boston University, 110 Cummington Mall, Boston, MA 02215, USA. Emails: tomwu@bu.edu, tron@bu.edu. The authors gratefully acknowledge the support by NSF award FRR-2212051.

Challenges. In addition to the difficulty brought by the kinematic constraints, solving the VIK problem is challenging for the following reasons:

1) Multiple camera poses that satisfy the field of view constraint may exist. However, due to the kinematic constraints of the robot, not all of the poses are kinematically feasible;

2) The pose of the frustum formed by the camera field of view is a nonlinear function of the joint configuration. Hence, restricting the object in this frustum yields a nonlinear constraint.

A naive approach would be to decouple the problem into two steps: (1) finding a camera pose that satisfies the field of view constraint, and (2) solving the inverse kinematics problem using this pose. However, this approach might fail due to reason *1)* alone. Another approach would be to incorporate the field of view constraint in the kinematics problem of the robot and solve it numerically. However, as mentioned in reason *2)*, the additional nonlinearity makes regular IK solvers prone to fail due to local minima.

Applications. A related area of research is visual servoing, which is the application of vision data for the feedback control of a robot [7]. Visual servoing has been applied to different robotic scenarios such as tracking and positioning of Unmanned Aerial Vehicles (UAVs) [8]–[10], ground robots [11]–[13], manipulators [14]–[16], and etc.

Because the control relies on vision data, it is vital to maintain the target in the Field of View. Prior works ensure visibility in VS via trajectory optimization [8], [9], [17], [18], control barrier functions [10], [16], [19], and learning-based control [13], [20].

Method summary. In this paper, we define and propose a way to solve the visual inverse kinematics problem. Unlike the aforementioned visual servoing techniques, the VIK problem seeks to find a configuration of the robot that satisfies the visibility and kinematic constraints. This configuration can serve as an initialization or target for visual servoing, similar to how IK is used in joint-based control. Another difference is that solving the VIK problem does not need to have actual image measurements (as VS does).

In this work, the visual constraint is modeled using a series of *virtual* links. Similar ideas have been seen in the domain of visual servoing. For example, *virtual linkage* is used in VS to connect the camera and the target such that the control tasks are similar to the ones of controlling robots subject to physical contacts with rigid bodies [21], [22]. In [23], measurements are given by images of legs of parallel robots, and the vision+kinematic system is reduced to a *hidden robot*

that, when controlled, achieves the desired task. Compared to the above relevant concepts in VS, our work does not use images of the robot, but we use a similar concept of modeling image projection via virtual robot links, and our goal is not to control, but to find a pose under which the virtual links satisfy specific joint angle constraints (representing field of view constraints).

We formulate the kinematic constraints of the robot using the method introduced in [24] for the VIK problem; this method uses a semidifinite programming (SDP) relaxation followed by a rank minimization technique on fixed-trace matrices. This method requires solving only a series of convex problems and has local convergence guarantees. While the original paper [24] considers only kinematic constraints such as joint axis and angle limits, an expanded algorithm adds prismatic joints in [25].

Paper contributions. This paper provides the following contributions:

- We introduce the VIK problem that fills the gap between inverse kinematics and visual servoing.
- We add the visibility constraints as a series of *virtual* prismatic joints. The visibility constraint is then relaxed to an SDP constraint and included together with the kinematic constraints to form the feasible set of the VIK problem.
- Along the way, we propose different vision-based costs to fulfill various objectives (e.g., matching feature positions to an image taken in advance).
- We use a solver from [24], [25]. However, while the formulation of that solver is very general, it was tested on relatively simple robots (arms, platform). As an additional contribution, in this paper, we test the solver in significantly more challenging configurations.

II. Kinematic constraints Using Rank-1 Semidefinite Matrices

This section reviews previous work [24] and [25] on how to model the kinematic constraints as the intersection of positive semidefinite matrices and rank-1 matrices. We start with a general formulation of the inverse kinematics problem which includes revolute and prismatic joints.

A. Revolute joints

We parameterize the robot kinematic chain using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ refers to the links and $\mathcal{E}$ refers to the joints. We denote $\{\mathbf{R}_i \in \mathbf{SO}(3), \mathbf{T}_i \in \mathbb{R}^3\}$ as the set of rigid body transformations from a reference frame attached to the link i to the world reference frame $\mathcal{W}$. The translation from frame i to a neighboring frame j, ${}^i\mathbf{T}_j$ is known, fixed, and related to the translations $\{\mathbf{T}_i\}$ and the rotation $\mathbf{R}_i$ by

$$\mathbf{T}_j - \mathbf{T}_i = \mathbf{R}_i {}^i\mathbf{T}_j. \tag{1}$$

Thus, we can parameterize the inverse kinematics problem as a function of the rotation matrices

$$\hat{\mathbf{R}} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{R}_2 & \dots & \mathbf{R}_i & \dots & \mathbf{R}_n \end{bmatrix}. \tag{2}$$

We denote a subset of the link indexes, $\mathcal{V}_r \subset \mathcal{V}$ such that each associated rotation $\mathbf{R}_i, i \in \mathcal{V}_r$ is unknown, and the corresponding matrix $\mathbf{R}$ as a truncated $\hat{\mathbf{R}}$ with only unknown rotations. We denote n_r as the number of unknown rotations.

As discussed in [24], the kinematic constraints of the inverse kinematics problem with revolute joints can be defined with the set $\mathcal{U}$, consisting of all vectors $\mathbf{u} = \mathrm{vec}(\mathbf{R}) \in \mathbb{R}^{9n_r}$ such that

$$\mathbf{A}_{\text{axis}}\mathbf{u} = \mathbf{b}_{\text{axis}}, \tag{3a}$$
$$\mathbf{A}_{\text{angle}}\mathbf{u} \leq \mathbf{b}_{\text{angle}}, \tag{3b}$$
$$\mathbf{R}_i \in \mathbf{SO}(3), \forall i \in \mathcal{V}_r, \tag{3c}$$

where (3a) and (3b) are linear constraints ensuring that the links sharing a revolute joint have a common axis and joint angle limit, respectively.

Remark 1: The constraints in (3) can be applied to spherical joints by removing the common axis constraint (3a). In this case, the angle limit constraint (3b) restricts a spatial conic limit of the two links.

The rotations matrices introduce the nonlinear constraints (3c). To prepare for the convex relaxation, for each $\mathbf{R}_i = \begin{bmatrix} \mathbf{R}_i^{(1)} & \mathbf{R}_i^{(2)} & \mathbf{R}_i^{(3)} \end{bmatrix} \in \mathbf{SO}(3)$, we define the variable

$$\mathbf{Y}_i = \begin{bmatrix} \mathbf{R}_i^{(1)} \\ \mathbf{R}_i^{(2)} \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_i^{(1)} \\ \mathbf{R}_i^{(2)} \\ 1 \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{7\times 7}. \tag{4}$$

There exists a linear transformation from $\mathbf{Y}_i$ to $\mathbf{R}_i$. The first two columns, $\mathbf{R}_i^{(1)}$ and $\mathbf{R}_i^{(2)}$, can be recovered from the last column of $\mathbf{Y}_i$ and the third column $\mathbf{R}_i^{(3)} = \mathbf{R}_i^{(1)} \times \mathbf{R}_i^{(2)}$ is a linear function of $\mathbf{Y}_i$. We denote $g(\cdot)$ as the transformation from $\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 & \mathbf{Y}_2 & \dots & \mathbf{Y}_i & \dots & \mathbf{Y}_{n_r} \end{bmatrix}$ to $\mathbf{R}$ such that $g(\mathbf{Y}) = \mathrm{vec}(\mathbf{R})$.

We define the following relaxation of $\mathcal{U}$.

Definition 1: The set $\bar{\mathcal{U}}$ is defined as all $\bar{\mathbf{u}} = g(\mathbf{Y}) \in \mathbb{R}^{9n_r}$ such that

$$\bar{\mathbf{u}} \text{ satisfies (3a) and (3b)} \tag{5a}$$
$$\mathbf{A}_{\text{structure}}\mathrm{vec}(\mathbf{Y}) = \mathbf{b}_{\text{structure}} \tag{5b}$$
$$\mathbf{Y}_i \succeq 0, \forall i \in \mathcal{V}_r \tag{5c}$$

where (5b) is a linear constraint that enforces the following structure of $\mathbf{Y}$ (resulted by combining (3c)) and (4))

1) $\mathrm{tr}(\mathbf{Y}_i(1:3, 1:3)) = \mathrm{tr}(\mathbf{Y}_i(4:6, 4:6)) = 1$;
2) $\mathrm{tr}(\mathbf{Y}_i(1:3, 4:6)) = 0$;
3) $\mathbf{Y}_i(7,7) = 1$.

The kinematic constraints $\mathcal{U}$ can be exactly captured by the relaxed set $\bar{\mathcal{U}}$ intersected with a rank-1 constraint, as detailed in the following theorem (see [24] for a proof).

Theorem 1: The set $\mathcal{U}$ is a subset of $\bar{\mathcal{U}}$, and is equal to the intersection of $\bar{\mathcal{U}}$ and $\mathcal{R}_1$, i.e., $\mathcal{U} = \bar{\mathcal{U}} \cap \mathcal{R}_1$, where $\mathcal{R}_1$ is the set of $\mathbf{u} = g(\mathbf{Y}) \in \mathbb{R}^{9n_r}$ such that each $\mathbf{Y}_i \in \mathbb{R}^{7\times 7}$ of $\mathbf{Y}$ is rank-1.

B. Prismatic Joints

The vision-based constraints use a formulation similar to that of the kinematic constraints for prismatic joints. A way to formulate such constraints is introduced in [25]. This subsection gives a brief review.

Let $\mathcal{E}_p \subset \mathcal{E}$ represent the set of prismatic joints and $\mathcal{V}_p$ the parents of prismatic joints. The prismatic joint $(i,j) \in \mathcal{E}_p$ can be defined by the following constraints.

Definition 2: The prismatic constraints are given by

$$\begin{aligned} &\mathbf{R}_i = \mathbf{R}_j \in \mathbf{SO}(3), \\ &\mathbf{T}_j = \mathbf{T}_i + (\tau_l + \tau_i(\tau_u - \tau_l))\mathbf{R}_i^{(3)}, \\ &\tau_i \in [0,1], \end{aligned} \tag{6}$$

where τ_l, τ_u are the lower- and upper-bound of the extension of the joint τ_i. The prismatic joint is assumed to be aligned along the z-axis in this definition.

To write convex constraints for (6), we define the variable $\mathbf{Y}_\tau = \begin{bmatrix} \mathbf{Y}_{\tau 1} & \mathbf{Y}_{\tau 2} & \dots & \mathbf{Y}_{\tau i} & \dots & \mathbf{Y}_{\tau n_p} \end{bmatrix}$, and

$$\mathbf{Y}_{\tau i} = \begin{bmatrix} \sqrt{\tau_i}\mathbf{R}_i^{(3)} \\ \sqrt{1-\tau_i}\mathbf{R}_i^{(3)} \\ \sqrt{\tau_i} \\ \sqrt{1-\tau_i} \end{bmatrix} \begin{bmatrix} \sqrt{\tau_i}\mathbf{R}_i^{(3)} \\ \sqrt{1-\tau_i}\mathbf{R}_i^{(3)} \\ \sqrt{\tau_i} \\ \sqrt{1-\tau_i} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{8\times 8}. \tag{7}$$

We define the constraint $\mathbf{Y}, \mathbf{Y}_\tau \in \bar{\mathcal{Y}}_\tau$ to restrict the following linear relations of $\mathbf{Y}_{\tau i}$ and $\mathbf{Y}_i$ entries: for $(i,j) \in \mathcal{E}_p$,

1) the trace of $\mathbf{Y}_{\tau i}$ equals 2;
2) $\mathrm{tr}(\mathbf{Y}_{\tau i}(1:3,1:3)) = \mathbf{Y}_{i\tau}(7,7)$ and $\mathrm{tr}(\mathbf{Y}_{\tau i}(4:6,4:6)) = \mathbf{Y}_{i\tau}(8,8)$;
3) $\mathbf{Y}_{\tau i}(4:6,7) = \mathbf{Y}_{\tau i}(1:3,8)$;
4) $\mathrm{tr}(\mathbf{Y}_{\tau i}(1:3,4:6)) = \mathbf{Y}_{\tau i}(7,8)$;
5) $\mathbf{Y}_{\tau i}(7,7) \in [0,1]$;
6) $\mathbf{Y}_{\tau i}(7,8) \geq 0$;
7) $\mathbf{Y}_{\tau i}(1:3,7) + \mathbf{Y}_{\tau i}(4:6,8) = \begin{bmatrix} \mathbf{Y}_i(2,6) - \mathbf{Y}_i(3,5) \\ \mathbf{Y}_i(3,4) - \mathbf{Y}_i(1,6) \\ \mathbf{Y}_i(1,5) - \mathbf{Y}_i(2,4) \end{bmatrix}$.

The above constraints restrict the structure of $\mathbf{Y}_{\tau i}$, its relation to $\mathbf{Y}_i$, and the bound on τ_i. These constraints are utilized in the following theorem (see [25] for a proof).

Theorem 2: Equations (6) hold if and only if

$$\mathbf{T}_j = \mathbf{T}_i + \tau_l \mathbf{R}_i^{(3)} + (\tau_u - \tau_l)\mathbf{Y}_{\tau i}(1:3,7), \tag{8}$$

$\mathbf{Y}_i,\ \mathbf{Y}_{\tau i} \in \mathbb{S}_+$, $\mathbf{Y}, \mathbf{Y}_\tau \in \bar{\mathcal{Y}}_\tau$, $\mathbf{Y}_i$ satisfies (5b), $\mathbf{R}_i = \mathbf{R}_j$, and $\mathrm{rank}(\mathbf{Y}_i) = \mathrm{rank}(\mathbf{Y}_{\tau i}) = 1$.

C. Inverse kinematics and rank-constrained optimization

Theorem 1 and 2 enables us to write (3) and (6) as a semidefinite constraint plus a rank-1 constraint on $\mathbf{Y}, \mathbf{Y}_\tau$. The IK problem can be formulated as the following

Problem 1 (Rank constrained inverse kinematics):

$$\begin{aligned} \min_{\mathbf{Y},\mathbf{Y}_\tau} \quad & f_{\text{ik}}(\mathbf{Y}, \mathbf{Y}_\tau) && \text{(9a)} \\ \text{subject to} \quad & g(\mathbf{Y}) \in \bar{\mathcal{U}}, && \text{(9b)} \\ & \mathbf{Y}, \mathbf{Y}_\tau \in \bar{\mathcal{Y}}_\tau, && \text{(9c)} \\ & \mathbf{Y}_i = \mathbf{Y}_j, \forall (i,j) \in \mathcal{E}_p, && \text{(9d)} \\ & \mathrm{rank}(\mathbf{Y}_i) = 1,\ i \in \mathcal{V}_r, && \text{(9e)} \\ & \mathrm{rank}(\mathbf{Y}_{\tau i}) = 1,\ i \in \mathcal{V}_p. && \text{(9f)} \end{aligned}$$

where f_{ik} is a quadratic function indicating the squared distance between the poses of the end-effector and the target. The constraints (9b) and (9e) ensure that $g(\mathbf{Y}) \in \mathcal{U}$ using Theorem 1. The constraints (9d), (9c), and (9f) together enforce the kinematic constraints for the prismatic joints defined in (6) hold using Theorem 2.

III. Visual Inverse Kinematics

This section introduces the visibility constraints along with different vision-based costs.

A. Perspective projection and virtual prismatic joints

In this work, we use the pinhole camera model [26]. We denote $\mathbf{R}_{\text{c}}$ as the rotation from camera axes X, Y, and Z to the world frame. A point $\mathbf{P} \in \mathbb{R}^3$ is projected to a point $\mathbf{p} \in \mathbb{R}^2$ on the image plane by the transformation $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$,

$$\mathbf{p} \sim \pi(\mathbf{P}) := \mathbf{C} \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix}, \tag{10}$$

where $\mathbf{C}$ is the camera matrix. To capture the point $\mathbf{P}$ in the field of view, $\mathbf{p}$ must be within the rectangular limit of the digital image. We approximate the field of view by restricting $\mathbf{p}$ to a circular region centered at the principal point with a radius of $h/2$, where h is the image height. In a 3-D world, the field of view then becomes a circular cone (assuming the height is infinitely large) whose apex is located at the camera center, and the axis is aligned with the camera Z-axis. We define α as the aperture (or half-angle) of the cone and $\alpha = r_\alpha \arctan(\frac{h}{2f_c})$, where f_c is the focal length and $r_\alpha \in (0,1]$ is a factor to control the tightness of the FoV constraint.

The visibility constraint ensures that the target stays in the field of view of the camera. To include this constraint in our formulation, we first model the robot and the filmed object using reference frames and then develop constraints by placing *virtual* joints between them. To begin with, we assume that the 3-D position of the object is available and that the object is characterized by a set of points $\mathcal{V}_o$, the position of which is referred to as $\{\mathbf{T}_i\}_{i\in\mathcal{V}_o}$ (for simplicity, we do not consider self-occlusion for the object). For each point $i \in \mathcal{V}_o$, we connect the point to the camera center (denoted as the index c) through a series of virtual joints: starting from the point, we put a prismatic joint followed by a spherical joint located at the camera center, as shown in Fig.1. Then we place two reference frames, attached to each link of the prismatic joint.

Remark 2: Ideally, the field of view should have the shape of a rectangular pyramid since images are rectangular.

However, the orientation of this shape is a function of the pose of the camera, which results in more involved FoV constraints. It should be possible to formulate also the constraint in our IK framework by introducing two additional coincident revolute joints at the camera center in addition to the prismatic joint. However, we decided to keep the scope of the paper focused on the simpler VIK problem with a conic FoV.

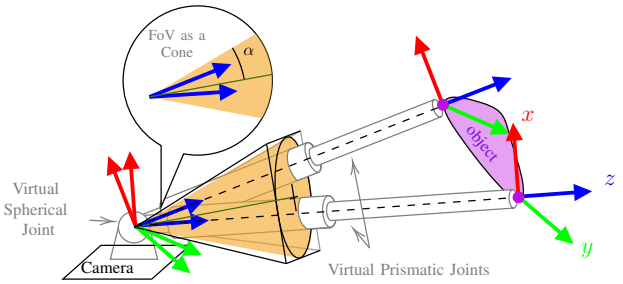


Fig. 1: The field of view is represented as a right circular cone whose axis is aligned with the camera Z-axis. The object is connected to the camera center through a virtual chain consisting of prismatic joints and spherical joints. Each prismatic joint is defined by two reference frames that share the same z-axis. Then, the FoV constraint is developed to restrict the z-axes in the cone.

B. Visibility constraint

From definition 2, the z-axes of the two frames attached to the links of the prismatic joint are aligned and pass through the camera center c and the point $i \in \mathcal{V}_o$. As a result, the field of view constraint is the same as restricting this z-axis in the cone, which is equivalent to enforcing the ball bound:

$$\mathbf{R}_i^{(3)} - \mathbf{R}_{\mathrm{c}}^{(3)} \in \mathcal{S}(\sqrt{2 - 2\cos(\alpha)}), \tag{11}$$

where $\mathbf{R}_i^{(3)}$ and $\mathbf{R}_{\mathrm{c}}^{(3)}$ are the third columns coincide with the z-axis of the two frames, $\mathcal{S}(r)$ is a ball centered at the origin with radius r. When $\mathbf{R}_i^{(3)}$ is on the cone surrounding $\mathbf{R}_{\mathrm{c}}^{(3)}$, the two unit vectors form an isosceles triangle with the apex angle equals α and legs equal 1. The length of the triangle base can be found as $2\sin(\alpha/2) = \sqrt{2 - 2\cos(\alpha)}$. It is clear that $\mathbf{R}_i^{(3)}$ remains in the cone when $\mathbf{R}_i^{(3)} - \mathbf{R}_{\mathrm{c}}^{(3)} \in \mathcal{S}(\sqrt{2 - 2\cos(\alpha)})$. This constraint is equivalent to the joint angle constraint (3b) in [24], where we restrict the difference of the first columns of the two neighboring reference frames in a ball.

We formulate the VIK problem similarly to what we do for the IK problem of the robot with additional virtual joints, which is

Problem 2 (Visual inverse kinematics):

$$\min_{\mathbf{Y}, \mathbf{Y}_\tau} \quad f_{\text{vik}}(\mathbf{Y}, \mathbf{Y}_\tau) \tag{12a}$$

$$\text{subject to} \quad \mathbf{Y}, \mathbf{Y}_\tau \text{ satisfy (9b)-(9f)}, \tag{12b}$$

$$\mathbf{A}_{\text{cone}}\mathbf{Y} \leq \mathbf{b}_{\text{cone}}. \tag{12c}$$

In Problem 2, the objective function (12a) is a vision-based cost, which we discuss in the next section. The constraint (12b) contains all the constraints in Problem 1 and captures the kinematics constraints of the robot while (12c) is a relaxation of the visibility constraint (11), where the ball restricting $\mathbf{R}_i^{(3)} - \mathbf{R}_{\mathrm{c}}^{(3)}$ is approximated with a polyhedron represented by a linear inequality constraints. Eventually, because $\mathbf{R}_i^{(3)} - \mathbf{R}_{\mathrm{c}}^{(3)}$ is a function of $\mathbf{Y}$, (12c) is obtained as a linear constraint on $\mathbf{Y}$.

C. Vision-based costs

The cost (12a) in Problem 2 can have different forms to fulfill different visual tasks. In this section, we introduce three costs that encode different objectives of the visual inverse kinematics problem. To perform flexible tasks, these costs can be used jointly by combining them linearly.

1) Levelness: To align the image with the ground, we minimize the deviation of the camera y-axis, $\mathbf{R}_{\mathrm{c}}$, from the world z-axis using the objective function

$$f(\mathbf{R}) = \|\mathbf{R}_{\mathrm{c}}^{(2)} - \begin{bmatrix}0 & 0 & 1\end{bmatrix}^{\mathsf{T}}\|_2^2. \tag{13}$$

When minimized, the cost indicates that the camera is close to an upright condition.

2) Centering: The visibility constraints keep the object in the image but do not specify *where* in the image. In some scenarios we would like to keep the object around the center of the image, motivating the following cost function.

$$f(\mathbf{R}) = \sum_{i \in \mathcal{V}_o} \|\mathbf{R}_i^{(3)} - \mathbf{R}_{\mathrm{c}}^{(3)}\|_2^2 \tag{14}$$

Minimizing (14) means that the vector in the l.h.s of (11) is not only bounded in the ball but also minimized for all the points $\mathcal{V}_o$. We can also see that minimizing (14) is equivalent to maximizing $\sum_{i \in \mathcal{V}_o} \mathbf{R}_{\mathrm{c}}^{(3)\mathsf{T}} \mathbf{R}_i^{(3)} = \sum_{i \in \mathcal{V}_o} \cos(\theta_{\mathrm{ic}})$, where θ_{ic} is the angle between the two vectors. Therefore by minimizing (14), the vectors $\{\mathbf{R}_i^{(3)}\}_{i \in \mathcal{V}_o}$ are averaged such that the vector $\mathbf{R}_{\mathrm{c}}^{(3)}$ is as close as possible to the mean on the unit sphere (interested readers are referred to [27] for averaging rotations on manifold).

Minimizing (14) is the same as maximizing $\sum_{i \in \mathcal{V}_o} \cos(\theta_{\mathrm{ic}})$, thus pushing the camera away from the points. We provide below another centering cost that, when minimized, yields the closest camera poses to the objects that center features in the image.

$$f(\mathbf{R}, \mathbf{T}) = \sum_{i \in \mathcal{V}_o} \|\mathbf{T}_i - \mathbf{T}_{\mathrm{c}} - \mathbf{R}_{\mathrm{c}}^{(3)}\|_2^2 \tag{15}$$

3) Minimizing the reprojection error: The reprojection error is the distance between the projection of a point $\mathbf{P}$ onto the image plane, $\mathbf{p}$, and a corresponding measurement $\hat{\mathbf{p}}$. Suppose we are given a preexisting image of the object and would like to find the configuration such that the reprojection error of the camera is minimized (e.g., as part of a standardization of an industrial inspection procedure). We denote the coordinates of the projections of the object points as $\{\hat{\mathbf{p}}_i\}_{i \in \mathcal{V}_o}$, and their corresponding 3-D vectors as $\tilde{\mathbf{p}}_i = \begin{bmatrix}\hat{\mathbf{p}}_i \\ f_c\end{bmatrix}$. The *spherical* reprojection error is given by

$$f(\mathbf{R}) = \sum_{i \in \mathcal{V}_o} \|\mathbf{R}_i^{(3)} - \mathbf{R}_c \frac{\tilde{\mathbf{p}}_i}{\|\tilde{\mathbf{p}}_i\|}\|_2^2, \tag{16}$$

where $\mathbf{R}_i^{(3)}$ is a unit vector pointing toward the point i and $\frac{\tilde{\mathbf{p}}_i}{\|\tilde{\mathbf{p}}_i\|}$ is the unit vector that passes through the corresponding point on the image. Minimizing (16) finds a solution such that the camera can take a picture that matches the given image.

D. Optimization

This subsection introduces a strategy to solve the visual inverse kinematics problem. At first, a convex relaxation of the problem is solved. After that, the solution is moved iteratively to the set of rank-1 matrices through a rank minimization algorithm based on the work in [25].

1) Convex relaxation: Without the visibility constraint (12c), Problem 2 reduces to a standard inverse kinematics problem. Therefore we can perform the same convex relaxation as we do for the robot inverse kinematics problem, which leads to the following formulation.

Problem 3 (Relaxed Visual inverse kinematics):

$$\min_{\mathbf{Y},\mathbf{Y}_\tau} \quad f(\mathbf{Y},\mathbf{Y}_\tau) \tag{17a}$$

$$\text{subject to} \quad \mathbf{Y},\mathbf{Y}_\tau \text{ satisfy (9b)-(9d)} \tag{17b}$$

$$\mathbf{A}_{\text{cone}}\mathbf{Y} \leq \mathbf{b}_{\text{cone}} \tag{17c}$$

The objective function (17a) can be any of the costs in Section III-C passed on to $\mathbf{Y},\mathbf{Y}_\tau$. Problem 3 is a relaxation of Problem 2 because they are the same except that the rank constraints (9e) and (9f), are omitted. We can apply an off-the-shelf solver to Problem 3 but the solution for $\mathbf{Y}_i$ and $\mathbf{Y}_{\tau i}$ is not, in general, rank-1. We briefly discuss below a way to project such solutions to the set of rank-1 matrices.

2) Rank minimization: The key idea in this rank minimization method is to maximize the largest eigenvalue of each $\mathbf{Y}_i$ and $\mathbf{Y}_{\tau i}$, and, because $\text{tr}(\mathbf{Y}_i)$ and $\text{tr}(\mathbf{Y}_{\tau i})$ are constant, the rest of the eigenvalues decrease accordingly and eventually render rank-1 matrices with only one non-zero eigenvalue. The method iteratively solves the following problem for an update $\{\mathbf{U}^k,\mathbf{U}_\tau^k\}$ of the variables $\{\mathbf{Y}^k,\mathbf{Y}_\tau^k\}$ at the k-th iteration.

Problem 4 (Update problem):

$$\min_{\mathbf{U}^k,\mathbf{U}_\tau^k} f(\mathbf{Y}^{k-1}+\mathbf{U}^k,\mathbf{Y}_\tau^{k-1}+\mathbf{U}_\tau^k) \tag{18a}$$

subject to

$$\sum_{i\in\mathcal{V}_r} \text{vec}(\mathbf{U}_i^k)^\mathsf{T}\nabla\lambda_1(\mathbf{Y}_i^{k-1}) \geq \sum_{i\in\mathcal{V}_r}(c-1)(\lambda_1(\mathbf{Y}_i^{k-1})-3) \tag{18b}$$

$$\sum_{i\in\mathcal{V}_p} \text{vec}(\mathbf{U}_{\tau i}^{k-1})^\mathsf{T}\nabla\lambda_1(\mathbf{Y}_{\tau i}^{k-1}) \geq \sum_{i\in\mathcal{V}_p}(c-1)(\lambda_1(\mathbf{Y}_{\tau i}^{k})-2) \tag{18c}$$

$$\mathbf{Y}^{k-1}+\mathbf{U}^k,\mathbf{Y}_\tau^{k-1}+\mathbf{U}\tau^k \text{ satisfy (17b), (17c).} \tag{18d}$$

In Problem 4, $\lambda_1(\cdot)$ is the largest eigenvalue as a function of a matrix, and $c\in[0,1]$ is a constant. The complete algorithm for solving the visual inverse kinematics problem is presented in Algorithm 1. This algorithm uses an adaptive approach in steps 4-7 to find a constant c that yields feasible solutions of Problem 4. By doing this, the algorithm dynamically selects a c that increases the largest eigenvalues [25, Sec. VII.D].

Algorithm 1 Visual Inverse Kinematics Solver

Input $\{\mathbf{T}_i\}_{i\in\mathcal{V}_o}$, $\mu,\epsilon_1,\epsilon_2,k_{max},p_{max},c_0,a$
Output $\{\mathbf{R}_i^*,\mathbf{T}_i^*\}_{i\in\mathcal{V}_r}$

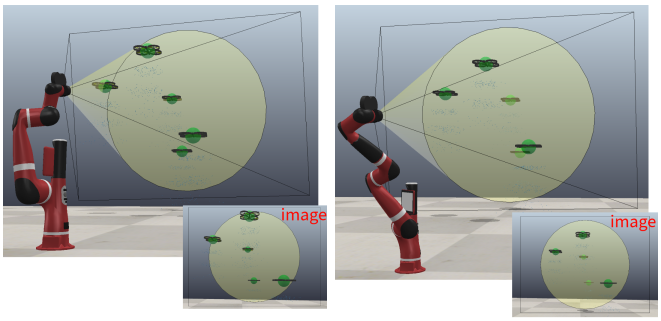
1: Solve Problem 3 to get an initial solution $\mathbf{Y}^0,\mathbf{Y}_\tau^0$ and set $k=1$, $p=1$.
2: **while** $(\exists\lambda_1(\mathbf{Y}_i)\leq 3-\epsilon_1 \;||\; \exists\lambda_1(\mathbf{Y}_{\tau i})\leq 2-\epsilon_1)$ & $\|\mathbf{U}^k\|_F\geq\epsilon_2$ & $k\leq k_{max}$ **do**
3: For each $\mathbf{Y}_i^{k-1}$ and $\mathbf{Y}_{\tau i}^{k-1}$, compute the largest eigenvalues $\lambda_1(\mathbf{Y}_i)$ and $\lambda_1(\mathbf{Y}_{\tau i})$, respectively.
4: **while** $p\leq p_{max}$ & Problem 4 is infeasible **do**
5: $c=1-(1-c_0)^{(a(p-1)+1)}$, $p\leftarrow p+1$.
6: Solve Problem 4 to get $\mathbf{U}^k$ and $\mathbf{U}_\tau^k$.
7: **end while**
8: Update $\mathbf{Y}_i^k=\mathbf{Y}_i^{k-1}+\mathbf{U}_i^k$ for all $i\in\mathcal{V}_r$ and update $\mathbf{Y}_{\tau i}^k=\mathbf{Y}_{\tau i}^{k-1}+\mathbf{U}_{\tau i}^k$ for all $i\in\mathcal{V}_p$ and set $k=k+1$, $p=1$.
9: **end while**
10: Recover the rotations $\{\mathbf{R}_i\}$ from $g(\mathbf{Y}^{k-1})$.
11: Recover the translations $\{\mathbf{T}_i\}$ using (1).

The constants c_0 and a are two parameters that control how the c value is increased when seeking a feasible solution.

Each iteration of Algorithm 1 minimizes an upper bound on $\sum_i\lambda_i$, which is a concave cost, thus ensuring local convergence. See [25] for a detailed proof of convergence.

IV. Simulation Results

To validate the proposed method, simulations are performed on a 7-degrees-of-freedom Sawyer manipulator, which is mounted with a camera on its hand and tasked to take a photo of some quadrotors. We assume that the positions of the quadrotors are known and each quadrotor is simplified as a point. We connect each point with the camera center through the virtual joints mentioned in Section III-A. The visibility constraint requires the camera to capture the quadrotors in the field of view, i.e., to restrict the points in the viewing frustum.



(a) Level (b) Level and center

Fig. 2: Our solver finds a posture of Sawyer subject to the visibility constraint of capturing the quadrotors in the camera field of view while targeting different vision-based objectives.

Fig. 2 shows two postures solved for an example where five quadrotors hover in front of the manipulator and the

objectives are set differently: one with levelness cost alone and the other with a combination of levelness and centering (function (14)) costs. As seen in Fig. 2, the quadrotors are restricted in the circular cone mentioned in Section III-A. As expected, both of the images are level with the ground while the quadrotors in the right image are closer to the center. The robot consists of 7 revolute joints and is added with 5 prismatic and 1 spherical *virtual* joints. The cone aperture $\alpha = 15.5°$. Fig. 3 depicts the computational process of the solution in Fig. 2a by showing the changes of the largest eigenvalues, λ_1, which as seen, is increased to the maximal values fixed by the traces $\mathrm{tr}(\mathbf{Y}_i) = 3$ and $\mathrm{tr}(\mathbf{Y}_{\tau i}) = 2$.

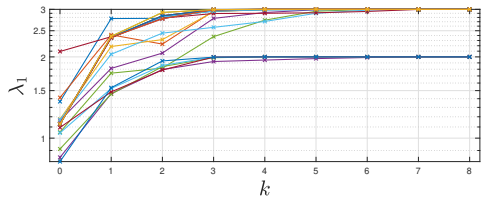


Fig. 3: The largest eigenvalues during the rank minimization process are increased to the fixed values of traces, i.e., $\mathrm{tr}(\mathbf{Y}_i)$=3 and $\mathrm{tr}(\mathbf{Y}_{\tau i})$=2, indicating rank-1 solutions. Each curve corresponds to the λ_1 of a matrix.

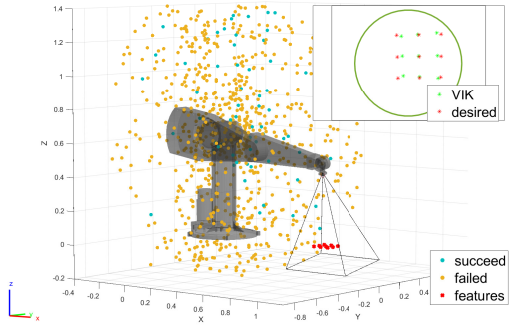


Fig. 4: Demonstration of how a random configuration fails to serve as a good initialization pose for visual servoing of a manipulator. Our method does not need an initial solution and finds a configuration such that the captured features are very close to the ones in the desired image.

We then show that our method can find an initial pose for eye-in-hand visual servoing algorithms. For comparison, we use randomly sampled camera poses to initialize the visual servoing. We start by placing 9 features to simulate a checkerboard placed in front of a PUMA 560 robot. Then, 1000 robot configurations are randomly selected within the task space. A standard gradient-based visual servoing algorithm from [28, Sec. V] is used to drive the camera from each initial pose to match with the image of a mutual pose that includes the features around the center of the image without considering FoV or joint limit constraints. The algorithm is counted as a failure if the error does not converge in 200 iterations. In the end, only 65 initial poses resulted in successful solutions. This shows that using a randomly generated initial pose can result in failures of VS. To efficiently initialize VS, we solve a VIK problem by minimizing the reprojection error (16) using Algorithm 1. The resulting configuration is presented in Fig. 4 along with the features of the desired image and those captured from the configuration solved by the VIK solver. It is seen that the features captured from the solved pose are close to the ones in the desired image. The difference arises from an increase in the cost function f in (16) (+0.0058) during the rank minimization process. Finally, the solved configuration is used to initialize VS and converge to the image pose successfully.

To evaluate our method's performance on various scenarios, we generate two sets of objects uniformly distributed in spaces $\{\mathbf{T}_i\}_{i\in\mathcal{V}_o}$ of two different sizes, marked as "condensed": $\begin{bmatrix}2.2 & -1.5 & -0.5\end{bmatrix}^\mathsf{T} \leq \mathbf{T}_i \leq \begin{bmatrix}6.8 & 1.5 & 6.5\end{bmatrix}^\mathsf{T}$ and "scattered": $\begin{bmatrix}1.4 & -2 & -1\end{bmatrix}^\mathsf{T} \leq \mathbf{T}_i \leq \begin{bmatrix}7.6 & 2 & 7\end{bmatrix}^\mathsf{T}$. Next, the generated sets of points are used as inputs in Algorithm 1 with different vision-based costs. The simulation is performed on Intel Core i7-10510U at 2.30 GHz CPUs with MOSEK [29] as solver. In this simulation, we use a quaternion parameterization introduced in [25, section VIII] as an additional technique to reduce the number of variables. It is shown in [25] that using this parameterization can improve the computation speed. Some other settings include choosing $c_0 = 0.1$ and $a = 4$ in the algorithm. To test how the method minimizes the reprojection error, we use solutions from tests with levelness plus centering costs to generate images. Then, we add random noise ($\leq 10^{-2}$) to the images and use our method to find solutions that match the features. The results are presented in Table I, where each row shows the results obtained from 500 tests, including success rate and for successful solutions: averaged computation time spent on solving the SDPs, averaged iterations taken, averaged cost increase $\Delta\bar{f}$ during the rank minimization algorithm, maximal distance of the rotation matrices $\{\mathbf{R}_i\}$ from their projections on $\mathbf{SO}(3)$, $\{P(\mathbf{R}_i)\}$, and the maximal second-largest eigenvalues of $\mathbf{Y}_i$ and $\mathbf{Y}_{\tau i}$.

It is seen in Table I that the method can recover matrices that are very close to real rotation matrices and can minimize rank to be very close to 1. In some simulations, the solver fails to find a solution within the loop in steps 4-7 in limited attempts p_{max}. This is because the algorithm guarantees only local convergence, and if started from a bad initial point, the algorithm can get stuck at solutions with rank higher than one. It is also observed that adding more points and expanding the distribution can reduce the success rate and slow down the computation, i.e., making the problem more difficult to solve. In general, the results in Table I show that the proposed method can find solutions to visual inverse kinematics problems that have various costs and inputs.

V. Conclusions

In this work, we introduce the visual inverse kinematics problem and formulate the camera visibility constraint as SDP constraints on a series of virtual joints. We provide multiple vision-based costs to fulfill different objectives. We

Distribution	Number of points	Type of cost	Avg. time / iterations	Success rate	$\Delta\bar{f}$	$\max(\|\mathbf{R}_i - P(\mathbf{R}_i)\|_F)$	maximal e_2
Condensed	5	①	1.4892(s)/8.1820	100%	0.1429	$2.7924 \cdot 10^{-4}$	$4.1864 \cdot 10^{-5}$
Condensed	5	①+②a	1.3432(s)/7.8380	100%	0.2749	$2.7250 \cdot 10^{-4}$	$3.2707 \cdot 10^{-5}$
Condensed	5	①+②b	1.5849(s)/6.4334	94.6%	0.3907	$2.7667 \cdot 10^{-4}$	$3.6456 \cdot 10^{-5}$
Condensed	15	①+②a	4.6871(s)/8.5991	91.8%	0.4491	$2.7834 \cdot 10^{-4}$	$2.3803 \cdot 10^{-5}$
Condensed	15	③	2.2349(s)/7.0331	98.7%	0.0178	$2.7050 \cdot 10^{-4}$	$4.0702 \cdot 10^{-5}$
Scattered	5	①	2.5394(s)/9.2511	87.6%	0.1770	$2.7953 \cdot 10^{-4}$	$6.4208 \cdot 10^{-5}$
Scattered	5	①+②a	1.6593(s)/8.4910	88.8%	0.3738	$2.7873 \cdot 10^{-4}$	$2.7337 \cdot 10^{-5}$
Scattered	15	①+②a	5.6191(s)/9.023	69.6%	0.7133	$2.8161 \cdot 10^{-4}$	$3.3635 \cdot 10^{-5}$
Scattered	15	③	2.3684(s)/7.0708	97.4%	0.0060	$2.5488 \cdot 10^{-4}$	$7.2774 \cdot 10^{-6}$

Types of cost are labeled as: ① levelness; ②a centering using (14); ②b centering using (15); ③ minimizing the reprojection error.

TABLE I: Performance of the proposed method on different visual inputs and vision-based costs. Each row shows results from 500 tests.

then provide a way to find solutions using a semidefinite relaxation followed by a rank minimization technique on fixed-trace matrices. For future work, we aim to investigate the cost increases observed during rank minimization in our simulations. Our goal is to develop post-processing strategies to mitigate these increases. Moreover, we plan to expand the proposed vision-kinematics framework to other applications in robotics such as camera pose estimation and robot hand-eye-calibration.

REFERENCES

[1] H.-Y. Lee and C.-G. Liang, "Displacement analysis of the general spatial 7-link 7r mechanism," *Mechanism and machine theory*, vol. 23, no. 3, pp. 219–226, 1988.

[2] M. Raghavan and B. Roth, "Inverse kinematics of the general 6r manipulator and related linkages," 1993.

[3] B. Kenwright, "Inverse kinematics–cyclic coordinate descent (ccd)," *Journal of Graphics Tools*, vol. 16, no. 4, pp. 177–217, 2012.

[4] T. Le Naour, N. Courty, and S. Gibet, "Kinematics in the metric space," *Computers & Graphics*, vol. 84, pp. 13–23, 2019.

[5] H. Dai, G. Izatt, and R. Tedrake, "Global inverse kinematics via mixed-integer convex optimization," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1420–1441, 2019.

[6] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," in *Computer graphics forum*, vol. 37, no. 6. Wiley Online Library, 2018, pp. 35–58.

[7] F. Chaumette and S. Hutchinson, "Visual servo control, I: Basic approaches," vol. 13, no. 4, pp. 82–90, 2006.

[8] M. Sheckells, G. Garimella, and M. Kobilarov, "Optimal visual servoing for differentially flat underactuated systems." IEEE, 2016, pp. 5541–5548.

[9] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, "Vision-based minimum-time trajectory generation for a quadrotor UAV," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6199–6206.

[10] D. Zheng, H. Wang, J. Wang, X. Zhang, and W. Chen, "Toward visibility guaranteed visual servoing control of quadrotor UAVs," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 1087–1095, 2019.

[11] R. Wei, D. Austin, and R. Mahony, "Biomimetic application of desert ant visual navigation for mobile robot docking with weighted landmarks," *International Journal of Intelligent Systems Technologies and Applications*, vol. 1, pp. 174–190, 2005.

[12] A. Cherubini, F. Chaumette, and G. Oriolo, "Visual servoing for path reaching with nonholonomic robots," *Robotica*, vol. 29, no. 7, pp. 1037–1048, 2011.

[13] Y. Wang, H. Lang, and C. W. De Silva, "A hybrid visual servo controller for robust grasping by wheeled mobile robots," *IEEE/ASME transactions on Mechatronics*, vol. 15, no. 5, pp. 757–769, 2009.

[14] D. Kragic, H. I. Christensen *et al.*, "Survey on visual servoing for manipulation," *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, p. 2002, 2002.

[15] A. Al-Shanoon and H. Lang, "Robotic manipulation based on 3-d visual servoing and deep neural networks," *Robotics and Autonomous Systems*, vol. 152, p. 104041, 2022.

[16] F. Dursun, B. V. Adorno, S. Watson, and W. Pan, "Maintaining visibility of dynamic objects in cluttered environments using mobile manipulators and vector field inequalities," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 6371–6378.

[17] C. Potena, D. Nardi, and A. Pretto, "Effective target aware visual navigation for UAVs," in *2017 European Conference on Mobile Robots (ECMR)*. IEEE, 2017, pp. 1–7.

[18] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[19] Y. Huang, M. Zhu, Z. Zheng, and K. H. Low, "Linear velocity-free visual servoing control for unmanned helicopter landing on a ship with visibility constraint," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 5, pp. 2979–2993, 2021.

[20] G. Fu, H. Chu, L. Liu, L. Fang, and X. Zhu, "Deep reinforcement learning for the visual servoing control of uavs with fov constraint," *Drones*, vol. 7, no. 6, p. 375, 2023.

[21] F. Chaumette, P. Rives, and B. Espiau, "Classification and realization of the different vision-based tasks," in *Visual servoing: real-time control of robot manipulators based on visual sensory feedback*. World Scientific, 1993, pp. 199–228.

[22] A. Castaño and S. Hutchinson, "Visual compliance: Task-directed visual servo control," *IEEE transactions on Robotics and Automation*, vol. 10, no. 3, pp. 334–342, 1994.

[23] S. Briot, P. Martinet, and V. Rosenzveig, "The hidden robot: An efficient concept contributing to the analysis of the controllability of parallel robots in advanced visual servoing techniques," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1337–1352, 2015.

[24] L. Wu and R. Tron, "An SDP optimization formulation for the inverse kinematics problem," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 4731–4738.

[25] ——, "IKSPARK: An inverse kinematics solver using semidefinite relaxation and rank minimization," *arXiv preprint arXiv:2403.12235*, 2024. [Online]. Available: https://arxiv.org/abs/2403.12235

[26] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[27] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International journal of computer vision*, vol. 103, no. 3, pp. 267–305, 2013.

[28] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[29] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.0.*, 2022.