

Convex Optimisation for Inverse Kinematics

Tarun Yenamandra^{1,2}, Florian Bernard^{1,2}, Jiayi Wang^{1,2}, Franziska Mueller^{1,2}, and Christian Theobalt^{1,2}
¹MPI Informatics, ²Saarland Informatics Campus

Abstract

We consider the problem of inverse kinematics (IK), where one wants to find the parameters of a given kinematic skeleton that best explain a set of observed 3D joint locations. The kinematic skeleton has a tree structure, where each node is a joint that has an associated geometric transformation that is propagated to all its child nodes. The IK problem has various applications in vision and graphics, for example for tracking or reconstructing articulated objects, such as human hands or bodies. Most commonly, the IK problem is tackled using local optimisation methods. A major downside of these approaches is that, due to the non-convex nature of the problem, such methods are prone to converge to unwanted local optima and therefore require a good initialisation. In this paper we propose a convex optimisation approach for the IK problem based on semidefinite programming, which admits a polynomial-time algorithm that globally solves (a relaxation of) the IK problem. Experimentally, we demonstrate that the proposed method significantly outperforms local optimisation methods using different real-world skeletons.

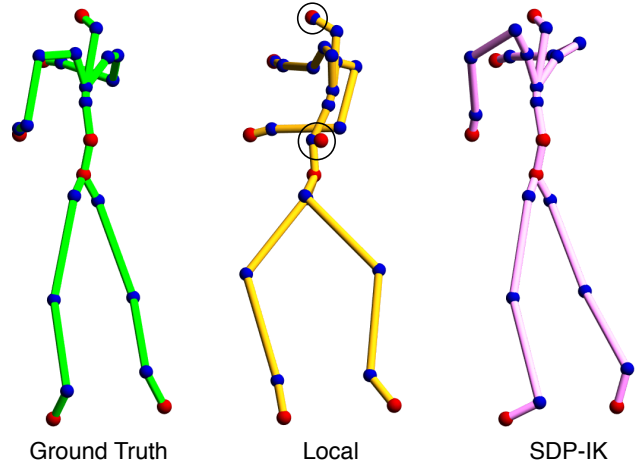


Figure 1. Our SDP-IK method results in a better fit compared to local optimisation. Left: Ground truth pose, where the observed points are shown in red. Centre: Local IK methods can get stuck in an unwanted local minimum. Right: Our convex SDP-IK formulation can reliably reach a good optimum that is much closer to the ground truth.

1. Introduction

The inverse kinematics (IK) problem plays an important role in robotics, computer games, graphics, and vision, as it is a fundamental building block for animating, controlling, tracking and reconstructing articulated objects, such as robotic arms or human bodies. The IK problem refers to the task of recovering parameters of a kinematic skeleton (e.g. joint angles), given a set of observed locations for (some of) the joints. The kinematic skeleton is represented as a tree, where each node is a joint that has translational and/or rotational degrees-of-freedom. In forward kinematics, one computes the positions of the joints given the translation and rotation parameters. Here, the transformation for each joint is defined relative to its parent joint, so that a chain of transformations is applied along the path from the root node to a leaf node (end-effector). Inverting this process, i.e. given some of the joint positions, one wants to recover the kinematic parameters that lead to this joint po-

sition configuration.

The IK problem is considered to be difficult due to several reasons. On the one hand, depending on the kinematic skeleton and the observed joint positions, the inverse kinematics can be ill-posed so that there may be multiple configurations of kinematic parameters that lead to the same joint positions. On the other hand, depending on the given joint locations, a solution that produces an exact fit to the observations might not exist. Lastly, the resulting optimisation problem is non-convex, which makes it generally difficult to find a globally optimal solution. Hence, approximations are oftentimes used in practice, which in turn require an initialisation that is sufficiently close to a global optimum. In computer vision, one of the most dominant applications of the IK problem is tracking and reconstructing articulated objects based on a temporal sequence of data (e.g. depth images, or RGB images). A common approach for tackling tracking applications is to initialise the kinematic parameters for the next frame using the tracked result from the current frame, and then solve the IK problem using local

optimisation methods. However, this is not possible for the very first frame of a sequence. Hence, in many works the authors assume that a good initialisation is available for the first frame, e.g. based on an initial calibration with a neutral pose, as done in [33]. In contrast, our proposed method is entirely initialisation-free, and is therefore well-suited for handling such cases. The main contribution of this work is a polynomial-time solution of the IK problem, coined *SDP-IK*, that finds a global optimum of a (convex relaxation of) the IK problem based on semidefinite programming.

2. Related Work

In this section we first address local optimisation methods for the inverse kinematics problem, followed by global methods. Subsequently, we summarise the most relevant works that consider semidefinite programming relaxations of related optimisation problems.

Local IK approaches: Local optimisation methods seek to iteratively find a solution of the IK problem based on a given initial estimate. One class of such methods use the first-order Taylor approximation of the problem and attempt to solve a linear system characterised by the Jacobian matrix. Alternatives are the Moore-Penrose pseudoinverse method, the Jacobian transpose method (equivalent to gradient descent for least-squares error), the Levenberg-Marquardt method (equivalent to gradient descent for damped least-squares error), and other variants [6, 10, 14, 20]. Second-order methods also exist, but such approaches require the computation of the Hessian of the forward kinematics function, which incur higher computational cost. Quasi-Newton methods, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, have been used to provide faster approximations for solving the IK problem [23].

Instead of trying to approximate and invert the forward kinematics function, heuristic methods employ simple rules to be iteratively followed and can often reach the IK solution. Cyclic coordinate descent (CCD) and its variants are heuristic methods that seek to minimise joint errors by changing one kinematic parameter at a time [27, 38]. Forward and backward reaching IK (FABRIK) [4] provides a method that can provably converge to the correct solution, when feasible, for a single unconstrained kinematic chain [3]. Evolutionary algorithms such as particle swarm optimisation and genetic algorithms, are heuristics inspired by evolution and are used to solve the IK problem in [34, 35].

Due to their speed and ease-of-use, local optimisation methods are widely used in applications such as character animation [20], motion re-targeting [19], model-based tracking [33, 36], post-processing on pose estimation [26, 28], and data visualisation [27]. However, despite their popularity, local IK approaches require a good initialisation for

complex, real-world skeletons, as otherwise they are prone to converge to unwanted local optima.

Global IK methods: Global IK methods aim to avoid this tendency getting stuck in local optima, and instead seek to obtain a global solution. One way of achieving initialisation independence is based on training a machine learning model to solve the IK problem. In [11], the authors learn an inverse kinematics function which maps from joint locations to kinematic parameters using a structured learning method.

An alternative to learning approaches are global optimisation methods. In [16], the authors tackle the problem of force-closure grasp synthesis based on sequential semidefinite programming, where rotation constraints are modelled in terms of bilinear matrix inequalities involving quaternions and rotation matrices. An approach for a feasibility formulation of the IK problem based on mixed-integer programming (MIP) is presented in [15]. The main idea here is to discretise all non-convex constraints based on binary variables. The resulting problem is then solved with a branch and bound algorithm, which is known to have exponential worst-case time complexity.

Contrary to the discussed works, which address feasibility versions of the IK problem, and/or do not admit polynomial-time algorithms, we propose a principled polynomial-time approach for the IK problem. Moreover, we consider a least-squares version of the IK problem, which is most relevant for the majority of applications in vision and graphics, such as for the tracking or the reconstruction of articulated objects.

Semidefinite programming relaxations: Semidefinite programming (SDP) relaxations are a popular way for tackling non-convex optimisation problems. Such methods have been successfully used for a range of different problems in vision and beyond, e.g. for graph matching [32, 40] or multi-graph matching [7, 21], the rigid registration of point-clouds [22, 25], the segmentation of images [39], or for permutation synchronisation [13]. However, generally such approaches are computationally expensive, since many of the SDP relaxations are based on a *lifting* of the variables, so that the size of the optimisation problem increases quadratically when moving from the original non-convex problem to the convex relaxation [21, 32, 40].

One scenario where SDP relaxations particularly stand out is in problems involving 3D rotation matrices. On the one hand, lifting a matrix variable of size 3×3 merely results in a relatively small variable of size $1 + (3 \cdot 3)^2 = 82$, so that such problems can be solved efficiently. On the other hand, some relaxations that involve a single rotation matrix have been observed to be tight in practice, i.e. even when solving a relaxation of a non-convex problem, the so-found solution is a global minimiser of the original non-convex problem. This has for example been empiri-

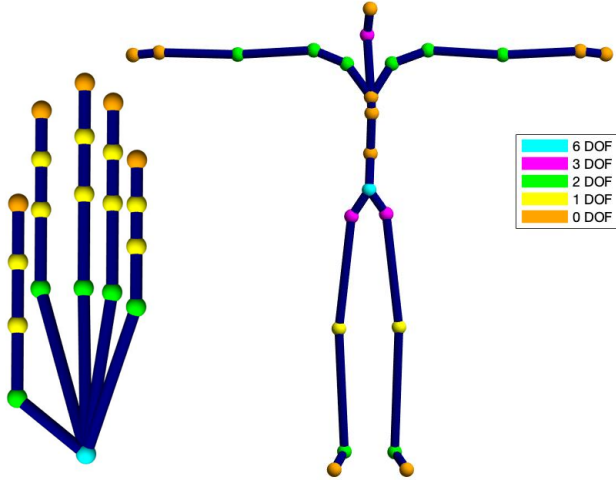


Figure 2. Illustration of a kinematic skeleton of a human hand (left) and a human body (right). The root joint for both skeletons admits 6 DOF (global translation and rotation), whereas all other joints admit between 0 and 3 rotational DOF.

cally demonstrated in [9] for the registration of 3D objects (with known correspondence). Other approaches that consider SDP approaches for problems involving rotations have been demonstrated for SLAM [30], pose-graph optimisation [12], or rotation averaging [5, 17, 37].

In our work we consider an SDP relaxation for the inverse kinematics problem, which involves a composition of several rotations that are propagated through the kinematic chain.

2.1. Notation

Here, we briefly outline the used notation. By \mathbf{I}_d we denote the $d \times d$ identity matrix, by $\mathbf{0}_d \in \mathbb{R}^d$ we denote the vector of all zeros, and the operator $\text{vec}(X)$ vectorises a given input matrix X by concatenating all the columns of X . For an integer n we use the notation $[n] := \{1, \dots, n\}$. For a matrix X we write $X_{:,i}$ to denote the vector that is formed by the i -th column of X , and analogously $X_{i,:}$ to denote the row vector that is formed by the i -th row of X . Moreover, for a 3D vector $x \in \mathbb{R}^3$, or a matrix $T \in \mathbb{R}^{3 \times 3}$, we use $\hat{x} = [x^T \ 1]^T \in \mathbb{R}^4$ and $\hat{T} \in \mathbb{R}^{4 \times 4}$ to denote their respective representation in homogeneous coordinates. For X being a matrix, the notation $X \succeq 0$ means that X is symmetric positive semidefinite.

3. Inverse Kinematics

In this section we describe our approach for tackling the inverse kinematics problem. To this end, we first define the forward kinematics model, which is followed by the precise statement of the problem.

3.1. Forward Kinematics Model

We assume that we are given a tree that defines the kinematic skeleton, see Fig. 2. The root of the tree has 6 degrees-of-freedom (DOF), of which 3 account for the global rotation, and 3 account for the global translation. Each non-root joint has between 0 and 3 rotational DOF, where each (non-zero) DOF accounts for a rotation of a given angle around a given axis. Joints without DOF are used to model the end-effectors, e.g. the fingertips in the human hand, cf. Fig. 2.

One-DOF representation: W.l.o.g., for convenience, we redefine this generic kinematic skeleton in such a way that each (non-end-effector) joint only has a single DOF that allows for a rotation around a given axis. To this end, for each joint of the original skeleton that has more than 1 DOF, we simply introduce one or two additional auxiliary joints (placed at the same position) that account for the additional DOF. We emphasise that while this redefinition of the kinematic skeleton does not change its kinematic behaviour, such a representation is more convenient for defining our SDP relaxation, as we will become apparent in Sec. 4.2.

Kinematic skeleton: Let $J \in \mathbb{N}$ be the total number of joints, where each joint now has at most 1 DOF (due to the redefinition of the skeleton). The global translation of the root is denoted by $t \in \mathbb{R}^3$. For each subsequent joint transformation, let $a_j \in \mathbb{R}^3$ for $j \in [J]$ denote the (unit-length) rotation axis of the j -th joint, and let $v_j \in \mathbb{R}^3$ be the “bone-vector” of the j -th joint, i.e. the offset of joint j in the coordinate system of its parent (cf. Fig. 2). For $\theta_j \in \mathbb{R}$ being the parameter of the j -th joint, by $\hat{T}(a_j, \theta_j) \in \mathbb{R}^{4 \times 4}$ we denote the transformation from the coordinate system of joint j to the coordinate system of its parent, represented in homogeneous coordinates. To be more specific, we have

$$\hat{T}(a_j, \theta_j) = \begin{bmatrix} K(a_j, \theta_j) & v_j \\ \mathbf{0}_3^T & 1 \end{bmatrix}, \quad (1)$$

where the rotation matrix $K(a, \theta)$ is obtained by Rodrigues’ rotation formula as

$$K(a, \theta) = \mathbf{I}_3 + \sin(\theta)[a]_{\times} + (1 - \cos(\theta))[a]_{\times}^2. \quad (2)$$

Here, $[\cdot]_{\times}$ is the skew-symmetric operator that generates a 3×3 matrix from a 3D vector (i.e. for $x, y \in \mathbb{R}^3$ we have $[x]_{\times} y = x \times y$). For joints with 0 DOF (i.e. for end-effectors), we define $K(a, \theta)$ to be the identity matrix \mathbf{I}_3 .

Forward model: For $\Theta := [t^T, \theta_1, \dots, \theta_J]^T \in \mathbb{R}^{3+J}$ being the parameter vector that stacks all joint parameters, the forward model for computing the position $x_j(\Theta)$ of the j -th joint is given by

$$\hat{x}_j(\Theta) = \hat{t} + \left[\prod_{i \in \alpha_j^1} \hat{T}(a_i, \theta_i) \right] \hat{\mathbf{0}}_3, \quad (3)$$

where by α_m^n we denote the path from the m -th joint to the n -th joint (from children to root) in the kinematic skeleton, and $\hat{\mathbf{0}}_3 = [0 \ 0 \ 0 \ 1]^T$ is the zero vector represented in homogeneous coordinates. For brevity, we will refer to all elements of Θ as *angles*, even if they represent translations.

Joint angle constraints: In addition, for each joint there is an interval \mathcal{I}_j that defines the range of valid values for θ_j so that it must hold that $\theta_j \in \mathcal{I}_j$. For notational convenience, we define $\mathcal{I} := \mathbb{R}^3 \times \mathcal{I}_1 \times \dots \times \mathcal{I}_J$ and write $\Theta \in \mathcal{I}$.

3.2. Problem Statement

We are interested in the problem of finding the parameters Θ such that the forward kinematics model best explains a given set of 3D joint position observations. Let $\mathcal{J} \subset [J]$ denote the subset of joints for which the 3D positions $y_j \in \mathbb{R}^3, j \in \mathcal{J}$, are known. The IK problem can now be phrased as a (constrained) nonlinear least-squares problem that reads

$$\begin{aligned} \min_{\Theta \in \mathcal{I}} \quad & \sum_{j \in \mathcal{J}} \|y_j - x_j(\Theta)\|^2 =: f(\Theta) \\ \Leftrightarrow \min_{\Theta \in \mathcal{I}} \quad & \sum_{j \in \mathcal{J}} \|\hat{y}_j - (\hat{t} + \left[\prod_{i \in \alpha_j^+} \hat{T}(a_i, \theta_i) \right] \hat{\mathbf{0}}_3)\|^2. \end{aligned} \quad (4)$$

Depending on the set \mathcal{J} , there may be multiple parameter vectors Θ that all lead to the same configuration of joint positions $x_j(\Theta)$ for all $j \in \mathcal{J}$. As such, for general IK problems of the form (4), the solution may not be unique since there can be multiple global optima. Most commonly, such problems are solved based on local optimisation methods, e.g. by modelling the hard constraints $\Theta \in \mathcal{I}$ as penalty in the objective function and then using a gradient descent procedure for locally optimising the objective function. A major downside of using such iterative approaches is that one requires a good initialisation for Θ , so that the optimisation does not result in an unwanted local optimum. We will tackle the problem of finding a good initialisation for Problem (4) based on a convex relaxation, as we will describe next.

4. Convex Relaxation for Inverse Kinematics

In order to achieve a convex relaxation of the IK problem, we will first redefine the problem as a non-convex quadratically constrained quadratic programme (QCQP) [1]. Subsequently, we will introduce our convex relaxation based on semidefinite programming.

4.1. Inverse Kinematics as QCQP

Rather than phrasing the IK problem in terms of the parameter vector Θ , we will directly optimise for rotation matrices and the global translation.

Global and relative rotations: Let $R_j \in \text{SO}(3) \subset \mathbb{R}^{3 \times 3}$ denote the *global* rotation of the j -th joint (i.e. relative to the root), and let $R_j^{\text{rel}} \in \text{SO}(3) \subset \mathbb{R}^{3 \times 3}$ denote the rotation of the j -th joint relative to its immediate parent, where we use the notation $\pi(j) \in [J]$ to indicate the parent of joint $j \in [J]$. For all joints we have the relation

$$R_j = R_{\pi(j)} R_j^{\text{rel}}, \quad (5)$$

where we define $R_{\pi(1)} = \mathbf{I}_3$. **Forward kinematics:** The position $p_j \in \mathbb{R}^3$ of the j -th (non-root) joint is defined recursively as

$$p_j = p_{\pi(j)} + R_{\pi(j)} v_j, \quad (6)$$

where $p_{\pi(j)} \in \mathbb{R}^3$ is the 3D position of its parent, and we define $p_{\pi(1)} := t$.

Joint angle constraints: For the j -th joint the rotation relative to its parent is constrained to be within the interval \mathcal{I}_j , see Sec. 3.1. In our reformulation we impose a similar constraint directly on the rotation matrix. In order to do so, we express R_j^{rel} in terms of a canonical rotation R_j^{can} , where in our case we choose (w.l.o.g.) a rotation around the x-axis, so that we have the general structure

$$R_j^{\text{can}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_j) & -\sin(\theta_j) \\ 0 & \sin(\theta_j) & \cos(\theta_j) \end{bmatrix} =: \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_j & -s_j \\ 0 & s_j & c_j \end{bmatrix}. \quad (7)$$

As such, we can write $R_j^{\text{rel}} = S_j^T R_j^{\text{can}} S_j$, for $S_j \in \text{SO}(3)$ being a suitably chosen matrix that is determined a priori (i.e. before optimisation). In order to impose the joint angle limits we enforce that c_j and s_j lie within the unit circle, i.e. we impose the convex constraint

$$c_j^2 + s_j^2 \leq 1. \quad (8)$$

In addition, for $\mathcal{I}_j = [\theta_j^-, \theta_j^+]$, we consider the line passing through $(\cos(\theta_j^-), \sin(\theta_j^-))$ and $(\cos(\theta_j^+), \sin(\theta_j^+))$, and enforce that the elements c_j and s_j of R_j^{can} are within a half-space defined by this line. This results in a linear inequality constraint in s_j and c_j . We use $R_j^{\text{can}} \in \mathbb{L}_j$ to refer to both of these (convex) joint angle constraints.

Rotation constraints: The set of (proper) rotations $\text{SO}(3)$ can be defined with quadratic constraints as

$$\text{SO}(3) = \{X \in \mathbb{R}^{3 \times 3} : X^T X = \mathbf{I}_3, X_{:,1} \times X_{:,2} = X_{:,3}\},$$

where the cross-product is used to implement the right-hand rule in order to ensure that the determinant is 1.

QCQP-IK: With the above elaborations we can now for-

mulate the IK problem as the QCQP

$$\begin{aligned}
& \min_{t, \{p_j, R_j, R_j^{\text{rel}}, R_j^{\text{can}}\}} \sum_{j \in \mathcal{J}} \|y_j - p_j\|^2 & (9) \\
\text{s.t.} \quad & p_j = p_{\pi(j)} + R_{\pi(j)} v_j, \quad p_{\pi(1)} = t, \\
& R_j = R_{\pi(j)} R_j^{\text{rel}}, \quad R_{\pi(1)} = \mathbf{I}_3, \\
& R_j^{\text{rel}} = S_j^T R_j^{\text{can}} S_j, \\
& R_j^{\text{can}} \in \mathbb{L}_j, \\
& (R_j, R_j^{\text{rel}}, R_j^{\text{can}}) \in \text{SO}(3)^3,
\end{aligned}$$

where the constraints are applied for all $j \in [J]$.

4.2. Inverse Kinematics as SDP

Before we introduce our semidefinite programming relaxation of the IK problem, we briefly summarise the main idea of semidefinite programming relaxations for general QCQPs.

4.2.1 Semidefinite Relaxations of Generic QCQPs

A generic QCQP can be written in canonical form as

$$\begin{aligned}
& \min_{x \in \mathbb{R}^n} x^T A_0 x & (10) \\
\text{s.t.} \quad & x^T A_i x \leq b_i,
\end{aligned}$$

where $A_i \in \mathbb{R}^{n \times n}$ are given symmetric matrices (that are possibly indefinite). Note that by using a homogeneous coordinate representation, this form also allows for linear terms in the objective as well as for linear constraints. Commonly, such non-convex QCQPs are solved by means of lifting, where an additional lifted variable Y of size $n \times n$ is introduced. Based on the property that for a given matrix A it holds that $x^T A x = \text{tr}(x^T A x) = \text{tr}(A x x^T)$, we can rewrite Problem (10) as

$$\begin{aligned}
& \min_{x \in \mathbb{R}^n, Y \in \mathbb{R}^{n \times n}} \text{tr}(A_0 Y) & (11) \\
\text{s.t.} \quad & \text{tr}(A_i Y) \leq b_i, \quad x x^T = Y.
\end{aligned}$$

It is well-known that the constraint $x x^T = Y$ is equivalent to

$$\begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \succeq 0, \quad \text{rank} \left(\begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \right) = 1.$$

Since the left part is a convex cone constraint, to obtain a convex relaxation the rank constraint (that accounts for the non-convexity) is dropped, which leads to the semidefinite programming problem

$$\begin{aligned}
& \min_{x \in \mathbb{R}^n, Y \in \mathbb{R}^{n \times n}} \text{tr}(A_0 Y) & (12) \\
\text{s.t.} \quad & \text{tr}(A_i Y) \leq b_i \\
& \begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \succeq 0.
\end{aligned}$$

4.2.2 Semidefinite Relaxation for IK

In order to obtain our semidefinite programming relaxation for the inverse kinematics problem, we systematically apply the elaborations in Sec. 4.2.1 to Problem (9). In the following we will elaborate on this. **Matrix multiplication constraints:** For three orthogonal matrices X, Y, Z of size $n \times n$, the matrix constraint $X = YZ$ can equivalently be written as

$$\begin{bmatrix} \mathbf{I}_n & X \\ X^T & \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} Y \\ Z^T \end{bmatrix} \begin{bmatrix} Y \\ Z^T \end{bmatrix}^T. \quad (13)$$

Similarly as above, by dropping rank constraints, we obtain a convex relaxation of this constraint as

$$(X, Y, Z) \in \mathbb{M} := \{(X, Y, Z) : \begin{bmatrix} \mathbf{I}_n & Y^T & Z \\ Y & \mathbf{I}_n & X \\ Z^T & X^T & \mathbf{I}_n \end{bmatrix} \succeq 0\}.$$

Rotation constraints: As indicated in Sec. 4.1, the $\text{SO}(3)$ constraint can be represented with quadratic equality constraints. A semidefinite relaxation of the constraint $R \in \text{SO}(3)$ is achieved by working with the vectorised R , i.e. $\text{vec}(R) \in \mathbb{R}^9$, and introducing a lifted variable \mathbf{R} of size 9×9 , as done in Sec. 4.2.1. The interested reader is referred to [9, 31], where further details about the lifting and the constraints can be found. We use the notation $(R, \mathbf{R}) \in \text{SO}(3)$, where $\text{SO}(3)$ is a convex set, to indicate that (R, \mathbf{R}) is a pair of variables that satisfy the lifted rotation constraints.

Our convex relaxation: We now state the convex relaxation of the inverse kinematics problem, which reads

$$\begin{aligned}
& \min_{t, \{p_j, R_j, R_j^{\text{rel}}, R_j^{\text{can}}\}, \{\mathbf{R}_j, \mathbf{R}_j^{\text{rel}}, \mathbf{R}_j^{\text{can}}\}} \sum_{j \in \mathcal{J}} \|y_j - p_j\|^2 & (14) \\
\text{s.t.} \quad & p_j = p_{\pi(j)} + R_{\pi(j)} v_j, \quad p_{\pi(1)} = t, \\
& (R_j, R_{\pi(j)}, R_j^{\text{rel}}) \in \mathbb{M}, \quad R_{\pi(1)} = \mathbf{I}_3, \\
& R_j^{\text{rel}} = S_j^T R_j^{\text{can}} S_j, \\
& R_j^{\text{can}} \in \mathbb{L}_j, \\
& (R_j, \mathbf{R}_j) \in \text{SO}(3). \\
& (R_j^{\text{rel}}, \mathbf{R}_j^{\text{rel}}) \in \text{SO}(3). \\
& (R_j^{\text{can}}, \mathbf{R}_j^{\text{can}}) \in \text{SO}(3).
\end{aligned}$$

Practical considerations: We solve our convex relaxation using the general purpose modelling tool Yalmip [24] that interfaces the MOSEK solver [2]. In order to get a tighter convex relaxation it helps to introduce convex relaxations of redundant constraints, e.g. for orthogonality one would use convex relaxations for both the constraints $X^T X = \mathbf{I}$ and $X X^T = \mathbf{I}$, see [9]. Once we have found a solution to Problem (14), we first project the obtained matrices onto the set $\text{SO}(3)$ by means of Singular Value Decomposition, and subsequently extract the joint angles Θ

directly from the projected matrices. The so-obtained joint angles Θ are not necessarily an optimum of the original IK objective (4). Hence we use the found Θ as initialisation for a trust region method as implemented in Manopt [8]. We refer to our overall approach as SDP-IK .

5. Experiments

In this section we experimentally demonstrate the benefits of our proposed approach based on two real-world kinematic skeletons. To this end, in Sec. 5.1 we first compare different local optimisation methods. Subsequently, in Secs. 5.2 and 5.3, we compare the best-performing local optimisation method to our convex relaxation approach, where the effectiveness of the proposed SDP-IK will become apparent. Since local IK methods are sensitive to initialisation, for every pose we uniformly sample multiple random initialisations (5 for the experiments in Sec. 5.1, and 20 for the experiments in Sec. 5.2 and 5.3) within the joint limit constraints, and then run local optimisation for each of the sampled initialisations. Note that we show the results obtained by *all* random initialisations in the comparisons. We measure the quality of the results based on the square-root of the normalised f in (4), which we denote as

$$\text{cost}(\Theta) = \sqrt{\frac{1}{|\mathcal{J}|} f(\Theta)}. \quad (15)$$

In our experiments, we use two real-world skeletons (*hand* and *human body*, see Fig. 2) in 100 different natural poses sampled from captured motion data sequences [18, 29]. The motion data is represented as a sequence of angle vectors Θ that animate the hand and body kinematic skeletons with predefined bone vectors $\{v_j\}$. We obtain the “observed joint positions” for each pose by computing the forward kinematics using the angle vector Θ . For the evaluation we consider two different cases:

1. The observed joint positions are noise-free, i.e. there exist kinematic parameters that yield an exact fit with an objective value of 0 (cf. Sec. 5.2).
2. The observed joint positions have added noise, i.e. the existence of kinematic parameters for an exact fit is not guaranteed (cf. Sec. 5.3). This is the common case for practical applications.

5.1. Local Optimisation Methods

To choose a representative local optimisation method as baseline, we compare four methods on the task of fitting the hand skeleton to observed joint locations when all joints are observed (noise-free). For tackling the IK problem with local optimisation methods it is common practice to convert it to an unconstrained optimisation problem, where the joint

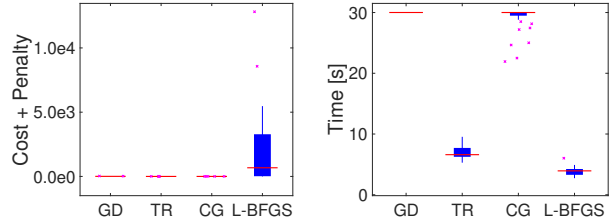


Figure 3. We show the performance of different local optimisation methods: gradient descent (GD), trust region method (TR), conjugate gradient method (CG), and L-BFGS.

angle constraints are modelled as penalties. As such, we obtain the differentiable optimisation problem

$$\min_{\Theta \in \mathbb{R}^{3+J}} \sum_{j \in \mathcal{J}} \|y_j - x_j(\Theta)\|^2 + \lambda \sum_{i=1}^{3+J} \text{dist}(\theta_i, \mathcal{I}_i)^2, \quad (16)$$

where $\text{dist}(\theta_i, \mathcal{I}_i)$ measures the distance of θ_i from the interval of plausible angles \mathcal{I}_i (as done in [28]), and λ is the hyperparameter that trades off joint limit violations with joint position errors. For our experiments we fix $\lambda = 100$.

In Fig. 3 we show results of gradient descent (GD), the trust regions method (TR), conjugate gradient (CG), and limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithms as implemented in the Manopt Matlab toolbox [8]. For all methods we used default parameters and allowed a maximum time budget of 30s. Each method is tested with the same 8 poses with 5 random initialisations, so that a total of 40 IK problems is solved. While the first three methods yield solutions with similar quality, L-BFGS performs significantly worse. Additionally, gradient descent and conjugate gradient are significantly slower compared to the other two methods. Hence we decide to use the trust region method (TR) as representative local IK method for the remaining experiments due to its good trade-off between speed and accuracy.

5.2. Fitting to Noise-Free Observations

In Fig. 4 we show quantitative results when solving the IK problem for noise-free observations. Our SDP-IK method almost always achieves an exact fit, i.e. the objective function value reaches 0, and thereby consistently outperforms the local optimisation approach. This occurs both when all joints are observed (**All**), and for the more ambiguous case where only the end-effectors and root are observed (**End+Spine** or **End+Wrist**), also cf. Fig. 2. Note that the local IK method suffers more from outliers due to bad initialisations, while our method can consistently find better minima.

5.3. Fitting to Noisy Observations

In many situations a given inverse kinematics problem can potentially be infeasible, i.e. kinematic parameters that

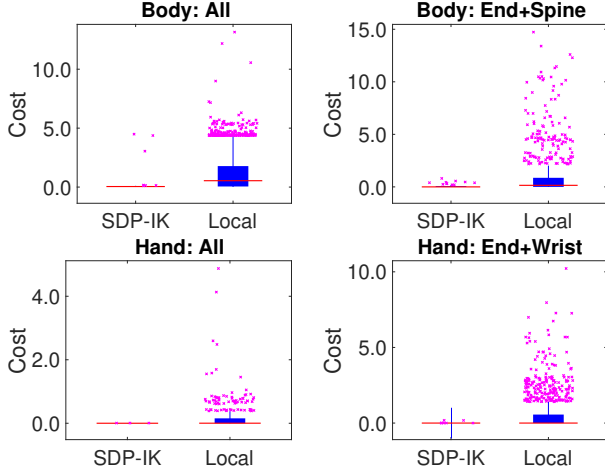


Figure 4. Box plot for fitting to noise-free observed joint locations of 100 poses for different skeletons (top: *body*, bottom: *hand*). Our SDP-IK method yields a consistently lower error in terms of cost (15), both in the case where all joints are given (top), as well as in the case when only the end-effectors are given (bottom).

exactly explain given observed joint locations may not exist. In practice, this is the case when the joint position observations are noisy. For example, neural networks have been employed successfully for joint location prediction for hands or human bodies. However, these networks mostly output a set of independent joint locations without any constraints to comply with the underlying kinematic structure. Hence, some approaches subsequently fit a kinematic skeleton to the predictions to optimise for plausible joint angles, usually relying on local IK methods like gradient descent. The aim then is to find kinematic parameters Θ and hence a kinematically consistent and plausible set of joint positions that achieve a minimum distance to the observed locations, as e.g. done in [26, 28].

In this experiment we mimic such a setting by fitting a kinematic skeleton to joint locations that exhibit *noise*, so that an exact fit may not be possible. To this end, we use the same poses as for the noise-free fitting in Sec. 5.2, and add noise to all observed joint locations. For each joint, the 3D noise offset $n = r \cdot d \in \mathbb{R}^3$ is obtained by uniformly sampling a unit length direction $d \in \mathbb{R}^3$, and a magnitude $r \in \mathbb{R}$ from $[0, r_{\max}]$, where r_{\max} is 10 mm for the hand skeleton, and 100 mm for the body skeleton.

For infeasible cases, the minimum value of the IK cost function in (15) fluctuates randomly with the sampled noise. The resulting box plot would merely capture how much each noisy sample violates the kinematic constraints instead of the quality of the solution. Hence, instead of reporting the raw value of the IK cost, we provide *normalised cost* values. To this end, for each pose we subtract the value of the smallest cost that is obtained by local optimisation among all the 20 random initialisations.

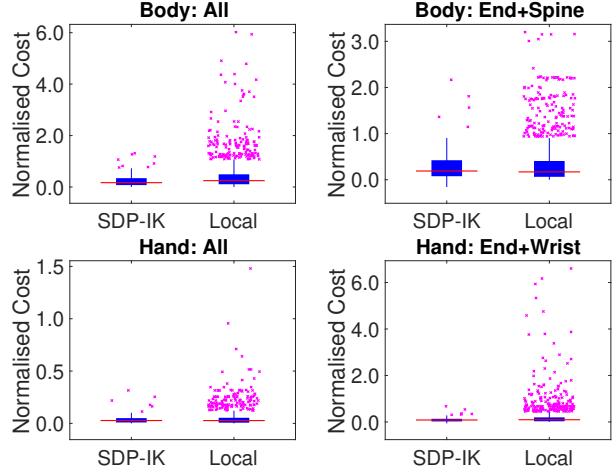


Figure 5. Box plot for fitting to noisy observed joint locations of 100 poses for different skeletons (top: *body*, bottom: *hand*). Our SDP-IK method yields lower or similar error in terms of the normalised cost, both in the case where all joints are given (left), as well as in the case when only the end-effectors are given (right).

In Fig. 5 we show that our method consistently achieves similar, or better performance than a local optimisation for the IK problem. Qualitative results for the local optimisation method, the projected solution of the semidefinite programming approach in (14), and our SDP-IK are shown in Fig. 6.

6. Discussion & Limitations

Currently, solving the semidefinite programming problem (14) takes about 17s on average on an Intel i5 CPU with 8GB of RAM, both for the hand as well as the human body skeleton. We emphasise that our current implementation is rather prototypical and is written so that it can work with generic kinematic skeletons. Moreover, we did not conduct any optimisations to improve the computational efficiency—we expect that runtime improvements by one order of magnitude or even more could be achieved using a problem-specific efficient implementation.

Currently, our model does not support translational DOF in non-root joints. In the future, we plan to also look into this case, as translational DOF can account for small changes in the bone length, which in turn can deal with the variance of bone lengths in real-world data.

7. Conclusion

We have presented a convex optimisation approach for the inverse kinematics problem based on a semidefinite programming relaxation. A major benefit of this approach is that we can find the global optimum of a (relaxation of the) IK problem, in contrast to local optimisation methods that heavily rely on good initialisations. Our experiments con-

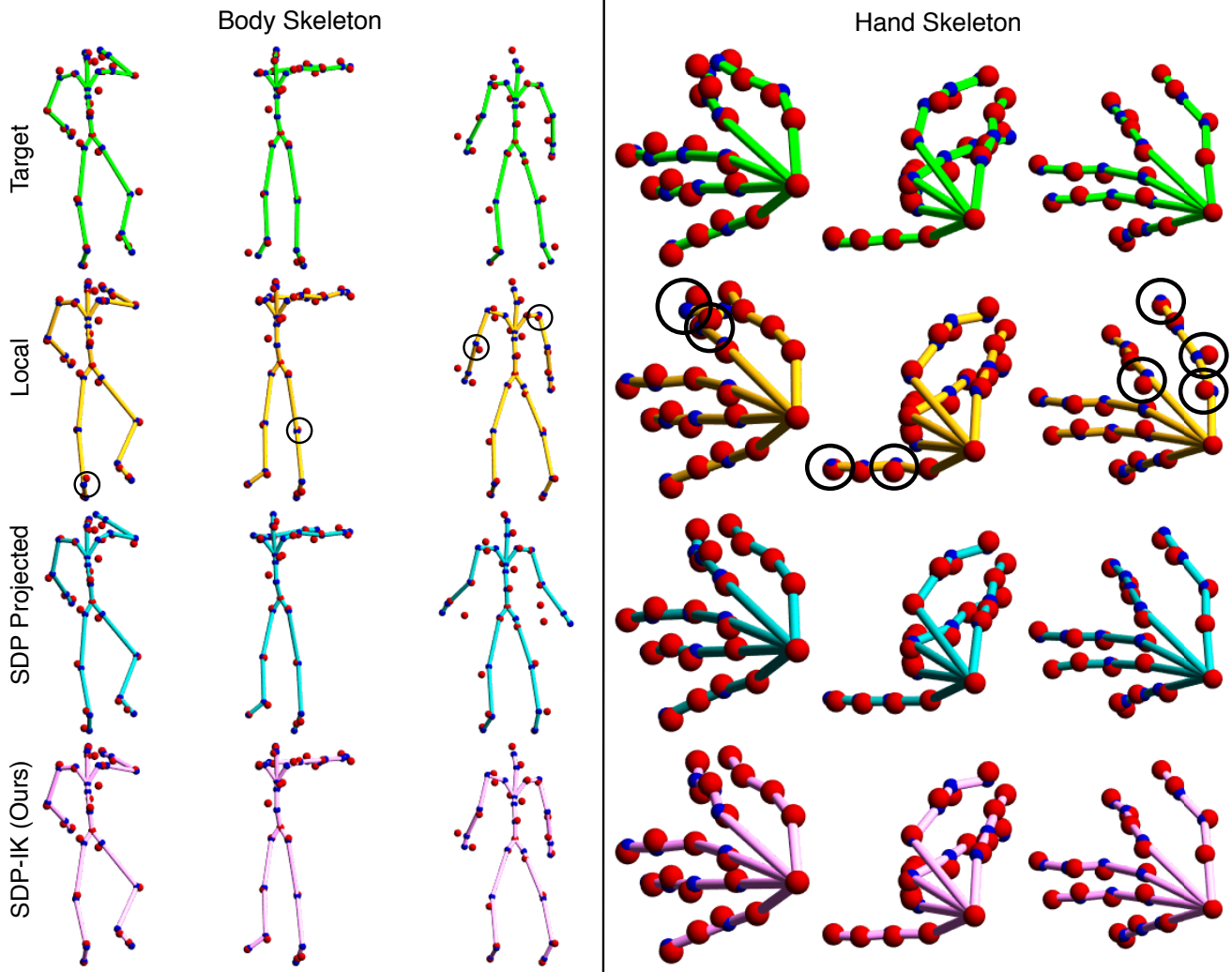


Figure 6. Qualitative results for fitting kinematic skeletons to known noisy points of human body and hand skeletons in different poses (red spheres: noisy points, blue spheres: fitted points). Top row: observed noisy joint positions. Second row: inverse kinematics solved via local optimisation using a random initialisation (erroneous joints are circled). Third row: solution of the semidefinite programming approach in (14) after projection. Bottom row: Our proposed SDP-IK method.

firm this advantage and also demonstrate that our proposed SDP-IK approach is a useful method for tackling inverse kinematics problems as they appear in computer vision and computer graphic problems.

Acknowledgements

This work was funded by the ERC Consolidator Grant 4DRepLy. We thank Marc Habermann for making the human body skeleton and motion sequences available to us.

References

- [1] K. M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43(2-3):471–484, 2009. 4
- [2] M. ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. 5
- [3] A. Aristidou, Y. Chrysanthou, and J. Lasenby. Extending fabrik with model constraints. *Computer Animation and Virtual Worlds*, 27:35–57, 02 2016. 2
- [4] A. Aristidou and J. Lasenby. FABRIK: A fast, iterative solver for the inverse kinematics problem. *Graph. Models*, 73(5):243–260, Sept. 2011. 2
- [5] A. S. Bandeira, M. Charikar, A. Singer, and A. Zhu. Multireference alignment using semidefinite programming. *ITCS*, 2014. 3
- [6] C. Bensalah, J. Gonzalez-Quijano, N. Hendrich, and M. Abderrahim. Anthropomorphic robotics hand inverse kinematics using estimated svd in an extended sdfs approach. In

- 2013 16th International Conference on Advanced Robotics (ICAR), pages 1–7, Nov 2013. 2
- [7] F. Bernard, C. Theobalt, and M. Moeller. DS*: Tighter Lifting-Free Convex Relaxations for Quadratic Matching Problems. In *CVPR*, 2018. 2
- [8] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. 6
- [9] J. Briales and J. Gonzalez-Jimenez. Convex Global 3D Registration with Lagrangian Duality. *CVPR*, 2017. 2, 5
- [10] S. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Transactions in Robotics and Automation*, 17, 05 2004. 2
- [11] B. Bcsi, D. Nguyen-Tuong, L. Csat, B. Scholkopf, and J. Peters. Learning inverse kinematics with structured prediction. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 698–703, Sep. 2011. 2
- [12] L. Carlone and G. C. Calafiore. Convex Relaxations for Pose Graph Optimization With Outliers. *IEEE Robotics and Automation Letters*, 2018. 3
- [13] Y. Chen, L. J. Guibas, and Q.-X. Huang. Near-Optimal Joint Object Matching via Convex Relaxation. In *International Conference on Machine Learning (ICML)*, 2014. 2
- [14] A. Colom and C. Torras. Redundant inverse kinematics: Experimental comparative review and two enhancements. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5333–5340, Oct 2012. 2
- [15] H. Dai, G. Izatt, and R. Tedrake. Global inverse kinematics via mixed-integer convex optimization. In *International Symposium on Robotics Research, Puerto Varas, Chile*, pages 1–16, 2017. 2
- [16] H. Dai, A. Majumdar, and R. Tedrake. Synthesis and optimization of force closure grasps via sequential semidefinite programming. In *Robotics Research*, pages 285–305. Springer, 2018. 2
- [17] A. P. Eriksson, C. Olsson, F. Kahl, and T.-J. Chin. Rotation Averaging and Strong Duality. *CVPR*, 2018. 3
- [18] M. Habermann, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt. Livecap: Real-time human performance capture from monocular video, 2019. 6
- [19] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard, and K. van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. In *ACM SIGGRAPH 2008 Papers, SIGGRAPH '08*, pages 27:1–27:11, New York, NY, USA, 2008. ACM. 2
- [20] B. Kenwright. Real-time character inverse kinematics using the gauss-seidel iterative approximation method. In *2012 4th International Conference on Creative Content Technologies (CONTENT 2012)*, pages 63–68, 2012. 2
- [21] I. Kezurer, S. Z. Kovalsky, R. Basri, and Y. Lipman. Tight Relaxation of Quadratic Matching. *Comput. Graph. Forum*, 2015. 2
- [22] Y. Khoo and A. Kapoor. Non-Iterative Rigid 2D/3D Point-Set Registration Using Semidefinite Programming. *IEEE Trans. Image Processing*, 2016. 2
- [23] Kwan Wu Chin, B. R. von Kinsky, and A. Marriott. Closed-form and generalized inverse kinematics solutions for the analysis of human motion. In *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 'Magnificent Milestones and Emerging Opportunities in Medical Engineering' (Cat. No.97CH36136)*, volume 5, pages 1911–1914 vol.5, Oct 1997. 2
- [24] J. Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *IEEE international conference on robotics and automation*, pages 284–289, 2004. 5
- [25] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4):73, 2016. 2
- [26] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. volume 36, 2017. 2, 7
- [27] D. Merrick and T. Dwyer. Skeletal animation for the exploration of graphs. In *Proceedings of the 2004 Australasian Symposium on Information Visualisation - Volume 35, APVis '04*, pages 61–70, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc. 2
- [28] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt. GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 6, 7
- [29] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2017. 6
- [30] D. M. Rosen, C. DuHadway, and J. J. Leonard. A convex relaxation for approximate global optimization in simultaneous localization and mapping. *ICRA*, 2015. 3
- [31] J. Saunderson, P. A. Parrilo, and A. S. Willsky. Semidefinite Descriptions of the Convex Hull of Rotation Matrices. *SIAM Journal on Optimization*, 2015. 5
- [32] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *EMMCVPR*, 2005. 2
- [33] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt. Real-time hand tracking using a sum of anisotropic gaussians model. In *Proceedings of the International Conference on 3D Vision (3DV)*, Dec. 2014. 2
- [34] S. Starke, N. Hendrich, S. Magg, and J. Zhang. An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1782–1789, 2016. 2
- [35] S. Starke, N. Hendrich, and J. Zhang. A memetic evolutionary algorithm for real-time articulated kinematic motion. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2473–2479, 2017. 2
- [36] A. Tkach, M. Pauly, and A. Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Trans. Graph.*, 35(6):222:1–222:11, Nov. 2016. 2

- [37] L. Wang and A. Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference*, 2013. [3](#)
- [38] L. . T. Wang and C. C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, Aug 1991. [2](#)
- [39] P. Wang, C. Shen, and A. van den Hengel. A Fast Semidefinite Approach to Solving Binary Quadratic Problems. *CVPR*, 2013. [2](#)
- [40] Q. Zhao, S. E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1):71–109, 1998. [2](#)