DMIK: A Highly General, Fast, Constrainable, and Stable Inverse Kinematics algorithm

This document introduces Delta Minimizing Inverse Kinematics (DMIK). DMIK is fast and stable and remains so under arbitrary joint orientation constraints, multiple end-effectors and intermediary effectors, position and orientation targets of arbitrary target weight priority, and any mix of the aforementioned. DMIK does not explicitly support telescoping joints, though whether it is amenable to is an open and unconsidered question.

Introduction and Motivation:

The two most popular general approaches to the Inverse Kinematics problem for real-time or semi real-time use cases are Cyclic Coordinate Descent (CCD) and Forward And Backward Reaching Inverse Kinematics (FABRIK). The strengths and weaknesses of these two approaches are almost precise complements of one another. Leaving developers to pick their poison.

CCD offers highly stable and fast solutions that behave well under joint constraints. However, CCD can solve only for single target-effector pairs, and becomes highly unstable when attempting to negotiate between multiple effectors with potentially mutually exclusive targets.

FABRIK offers highly stable and fast solutions that naturally handle multiple effectors with potentially mutually exclusive targets. However, FABRIK is, in practice, extremely unstable when applied to bones with joint constraints.

Foregoing joint constraints to ensure stable FABRIK solutions results in highly unnatural (often extremely painful) looking poses. While foregoing consideration of more than one target-effector pair per bone to ensure stable but well constrained CCD solutions results in incomplete poses, where even if two separate chains with two separate effectors could theoretically both reach their targets while obeying all constraints, only one effector actually does.

The Basic Idea:

DMIK can be thought of as a generalization of the concept (though not necessarily the mathematics) of Inverse Kinematics by CCD. However, where for every bone in a chain, CCD seeks to minimize the angular distance between the chain's end-effector and its corresponding target from the perspective of that bone, DMIK instead seeks to minimize the average distance between all effectors and their corresponding targets from the perspective of that bone.

More specifically, the DMIK procedure starts from the outermost bones of an armature, and proceeds rootward as follows:

1. Translate all relevant effectors and corresponding targets such that they are centered about the bone's origin.
2. Find the rotation that minimizes the average of the distances between each effector-target pair, and apply that rotation to the bone.
3. Rectify the bone's orientation to be back within an allowable orientation as per any limits imposed by any dampening parameters or joint constraint on the bone if necessary, then proceed to the next bone.

Once the root bone or effector has been reached, repeat this process until convergence or budget exhaustion.

A Deeper Look:

At first glance, the devil might appear to be hidden in step 2. But on closer consideration, step 2 amounts to a simple case of a *Constrained Orthogonal Procrustes Problem*. This type of problem arises often in bioinformatics, astronomy, crystallography and computer vision, and can be readily solved by any number of existing algorithms. Of these, the Kabsch alignment algorithm and the Quaternion Characteristic Polynomial (QCP) algorithms are perhaps the most popular. DMIK has been verified to work fine with either algorithm, but QCP is recommended for its speed, simplicity, framework agnosticism, and relatively few edge-cases[1].

Efficient implementations of both the QCP and Kabsch alignment algorithms are widely available in a number of languages, and as they (and related algorithms) are roughly interchangeable for our purposes, their mathematical peculiarities will not be covered, and the rest of this document will refer to whatever algorithm you choose for the purpose of minimizing the average of the distances of point-pairs as *The Minimizer*.

Orientation Targets:

Consider a humanoid (perhaps yourself, if you happen to be approximately humanoid in shape) sitting at a table, with the back of its hand flat against the table such that its fingers are pointing directly away from its torso. If the humanoid were instructed to rotate its hand around its middle knuckle, such that the back of its hand remained flat against the table, but its fingers now pointed toward its torso, the orientation of all of the bones in the humanoid, from its wrist up to its shoulder and perhaps even part of its spine would have to change drastically to allow for this.

[1] Note that with single floating point precision, QCP may under some degenerate conditions require renormalization of the resultant quaternion. This does not affect solution quality, but may be worth considering for anyone looking to hyper-optimize performance.

If we treat the humanoid’s pelvis as one effector and the chair as that effector’s target, and treat its knuckle bone as another effector, and the spot on the table to which the knuckle bone must remain affixed as the knuckle bone’s target, we see that even if the positions of the targets do not change at all, there can be massive differences in the poses an armature must adopt based solely on the orientations of its targets.

From this we see the importance of treating target orientations as first class citizens throughout the entire IK procedure. If we solve only for positions and leave target orientations as an afterthought (as CCD and FABRIK implementations most often do) we are left to decide between “cheating” and violating joint constraints so an effector is still aligned with its target (often resulting in effector joints that look like they’re painfully hyperextending), or else obeying the joint constraints and failing to solve for a completely valid target.

One of DMIK’s strengths is that almost no distinction at all is made by the solver between position and orientation targets. Both orientation and position are encoded as point pairs for the Minimizer to solve for.

This is done by representing each target as a set of up to seven points. One point representing the target origin, three points representing the basis vectors emanating from that origin, and three points representing the inverses of the basis vectors with respect to the target origin. Effectors are given the same treatment. These 7 effector-target pairs are then fed to the minimizer, which attempts to find the rotation that minimizes the average distance between all effector-target pairs.

Figure 1.1 - 1.2: Effector bases are marked in orange, target bases are marked in blue. Dashed magenta lines indicate the deltas between each effector basis vector/origin and its corresponding target basis vector / origin. ⊕ indicates the origin of the bone under consideration at this step in the algorithm.

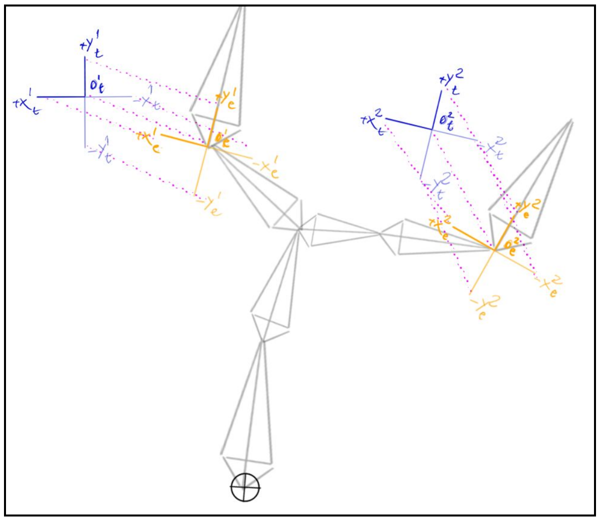

Figure 1.1 (left): *Armature prior to rotation about ⊕ so as to minimize average distances between effector points and corresponding target points.*

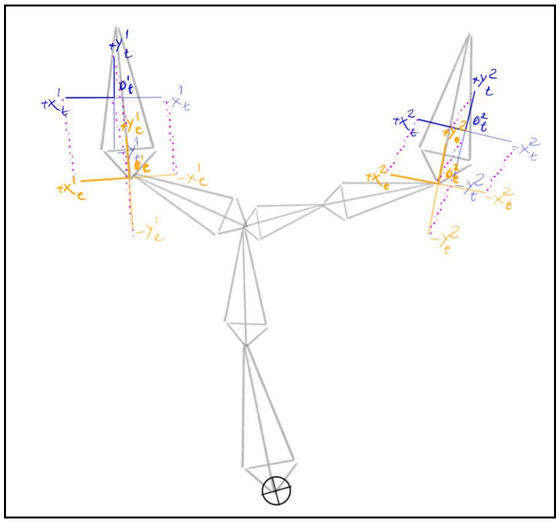

Figure 1.2 (right): *Armature* after rotation about ⊕ so as to minimize average distances between all descendant effector points and their corresponding target points.

Multiple End and Intermediary-Effectors:

SInce the Minimizer operates on point-pairs, solving for multiple effectors is similarly trivial. We just feed the Minimizer all additional effector-target point pairs for any other effectors we wish to optimize a bone's orientation for. Since the Minimizer optimizes for the minimum deviation between effector-target pairs, we can even weigh some targets more strongly than others by just scaling the effector-target pairs about the bone origin by an amount commensurate with the precedence we want to place on that target-effector pair. This works because rotation of any point closer to the origin results in a smaller change in euclidean distance than does rotation of any point farther from the origin.[2]

Constraints:

In theory, DMIK should work well with any type of commonly used joint constraint (doing so requires no more than implementing step 3 in the introductory section). Unfortunately, in practice, most commonly used joint constraints come with their own set of tradeoffs. While the problems presented by these tradeoffs aren't specific to

[2] It should be noted that this can happen incidentally, where one set of target-effector point-pairs happens to be closer to the origin of a bone being solved for than another set of target-effector point-pairs. In theory it might be prudent to adopt a normalization scheme to account for this if the Minimizer doesn't do so already. In practice, it doesn't really matter that much.

Inverse Kinematics, they are nevertheless best avoided. An ideal joint constraint system would be

1. Continuous: No sharp concave corners for a bone to get “stuck” in.
2. Versatile: It should be possible to specify any conceivable orientation region.
3. Expressive: The desired shape of the allowable orientation region should be easily and fully specifiable with as few parameters as possible.
4. Extensible: The constraints should be amenable to specification of any number of additional properties that may vary continuously throughout or beyond the allowable orientations region.
5. Fast:: as few operations as possible should be required to determine if a Bone is within the valid orientation region, or to determine the smallest rotation that brings it back within a valid orientation.