Inverse Kinematics for Atlas's Arms

Wyatt Newman
September, 2014

Introduction: Inverse kinematics for robots is the problem of finding a set of joint angles that results in satisfying some endpoint specification. Commonly, one may have a desired pose for an end effector (e.g. hand), specified fully in position and orientation (a "frame"). A full frame specification requires a minimum of 6 parameters. If the robot has fewer than 6 joints that can be used, then it is typically impossible to find a set of joint angles that satisfy the full set of output constraints. If there are more than 6 joints available, typically there are infinitely many solutions. If there are exactly 6 joints available, there is typically a countable number of solutions (between 0 and 8). However, these typical cases are frequently violated. If the target pose is out of reach, then there will be zero solutions, regardless of how many joints are available. Further, constraints on achievable joint angles can overconstrain the problem such that there are no viable solutions. In general, inverse kinematics is problematic.

For Atlas' arms (as of September, 2014), each arm has 6 joints. However, the configuration and the ranges of motion of these joints is such that it is very rare that Atlas can reach a specified hand pose. This problem is expected to be relieved with an anticipated arm replacement, in which the new arms will have 7 joints each. However, analysis of inverse kinematics for the 6-DOF (Degree Of Freedom) case is useful, as it can be applied to the current robot and simulator, and it is potentially extensible to future 7-DOF arms.

These notes augment the paper "Manipulation Planning for the Atlas Humanoid Robot" [1], filling in the inverse-kinematic solution details omitted there. (This paper is unpublished at the time of this writing, but will be made available with course materials).

Arm Specifications: The following table repeats the arm specifications given in the above-referenced paper.

TABLE DENAVIT HARTENBERG PARAMETERS FOR RIGHT ARM

Frame transform	*d*	*a*	*α*	*θ*
Torso to usy	0.079	0.052	7π/6	0
Usy to shx	-0.5172	0.0235	π/2	q1
Shx to ely	0	0.016	π/2	q2
Ely to elx	-0.306	0.0092	-π/2	q3
Elx to uwy	0	0.0092	-π/2	q4
Uwy to mwx	-0.306	0.0092	-π/2	q5
Mwx to flange	0	0	π/2	q6

Notably, the "a" parameters are relatively small, motivating the approximation to ignore them. Two other parameters define the length of the humerus, $L_{humerus}=0.306$, as well as the length of the forearm (which is the same length): $L_{forearm}=0.306$.

In addition to the above frames, a grasp frame is defined on the palm of the Sandia hand, which is useful for specifying power grasps. This frame has its origin on the surface of the palm, the z axis normal to the palm (pointing outwards), and the y axis lying parallel to a cylindrical "hump" feature (pointing towards the thumb). The transform between the "flange" frame and the defined "grasp" frame is hard coded as a set of constants in a 4x4 transformation matrix, ${}^{grasp}T_{flange}$. Note that the first line of the D-H table is also comprised of all constants (no variable q angle), and thus the transform ${}^{usy}T_{utorso}$ is also static (i.e., a set of constants).

As noted in [1], there is no known analytic inverse-kinematic solution for Atlas' arms. However, an approximately solution can be derived by ignoring the "a"-parameter offsets, which are relatively small. Importantly, the first three "shoulder" joints nearly intersect. We can thus define a approximate, corresponding system for which the a-parameters are set to zero, solve the inverse kinematics problem for this alternative system, and use the result for intuition with Atlas or as seed values to obtain precise solutions using an iterative, numerical algorithm.

Defining Shoulder, Elbow and Wrist Points: If the first three joints intersect, then we can define this intersection point as the "shoulder point", **s**. Further, we can define a wrist point, **w**, which lies at the intersection of the tool-flange z-axis and the wrist-bend joint axis (assuming there is no offset between these axes). We further assume that the forearm joint intersects both the wrist-bend joint and the elbow joint, and that the humerus joint axis also intersects the elbow-joint axis. Under these assumptions, we can define an elbow point, **e**, at the intersection of these three axes.

Numerically, we compute the values of **w** and **s** as follows. A desired grasp pose can be specified as a 4x4, ${}^{utorso}T_{des\ grasp}$, which specifies the desired grasp-frame origin and orientation, as expressed in the utorso coordinate frame. The transform ${}^{flange}T_{grasp}$ is defined as a static transform (i.e., comprised of defined, fixed values), expressing the grasp frame with respect to the flange frame. These can be combined to compute the wrist point as follows. Define:

$$ {}^{utorso}T_{flange} = {}^{utorso}T_{des\ grasp}\ {}^{flange}T_{grasp}^{-1} $$

Since the right-hand side above is comprised of all known values, the left-hand side 4x4 constitutes a fully-known expression for the desired location of the flange frame with respect to the utorso frame. The fourth column of this 4x4 contains the origin of the flange frame, which is the wrist point **w**. We thus have that the values of **w** follow from the desired grasp and defined constants.

Next, the shoulder point is computed by forward kinematics from the utorso frame to the shx frame. The first transform, ${}^{utorso}T_{usy}$, is a set of constants, computed as:

$$ {}^{utorso}T_{usy} = \begin{bmatrix} 1 & 0 & 0 & 0.052 \\ 0 & \cos(7\pi/6) & -\sin(7\pi/6) & 0 \\ 0 & \sin(7\pi/6) & \cos(7\pi/6) & 0.079 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The next transform, from usy to shx, depends on angle q1 (the usy angle). This transform is expressed as:

$$ {}^{usy}T_{shx} = \begin{bmatrix} \cos(q_{usy}) & -\sin(q_{usy}) & \sin(q_{usy}) & 0.0235\cos(q_{usy}) \\ \sin(q_{usy}) & 0 & -\cos(q_{usy}) & 0.0235\sin(q_{usy}) \\ 0 & 1 & 0 & -0.5172 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

However, if the offset a1 (=0.0235) is ignored, then the fourth column above is simplified. The shx frame can be defined with respect to the utorso frame as the product:

$$ {}^{utorso}T_{shx} = {}^{utorso}T_{usy}\ {}^{usy}T_{shx} $$

The approximate coordinates of the shoulder point are found from the fourth column of ${}^{utorso}T_{shx}$.

With the approximation that a1 =0 (i.e., ignoring the constant 0.0235), the 4th column of ${}^{utorso}T_{shx}$ contains:
$\left[0.052\,;0.5172\sin(7\pi/6),-0.05172\cos(7\pi/6)+0.079\right]$,
and these coordinates define the shoulder point, **s**.

Solution for Elbow Angle: The distance from **w** to **s** is a known value, computed as: $L_{reach}=\|w-s\|$. If the required reach is longer than the length of the humerus plus the length of the forearm, i.e. if $L_{reach}>L_{humerus}+L_{forearm}$, then there is no solution, and the IK algorithm should stop here, indicating no solutions exist.

Alternatively, if the goal is within reach, then the elbow angle can be computed from a law of cosines. The points **w, e** and **s** form a triangle. We can define the vector from **s** to **e** as $r_{e,s}$, and the vector from elbow to wrist as $r_{w,e}$. The lengths of these vectors are $L_{humerus}$ and $L_{forearm}$, respectively. The elbow angle follows from:
$L_{reach}^2=L_{humerus}^2+L_{forearm}^2-2\mathrm{L}_{humerus}L_{forearm}\cos(q_4)$.

For Atlas' right arm, only negative elbow angles are possible, so for the two solutions to the above equation, the negative solution is the only candidate for the elbow angle.

This solution should be tested against the range of physically-possible elbow angles, and if this solution is outside the range, the algorithm should return that no IK solutions exist.

If the elbow-angle solution is feasible, there may be multiple IK solutions, but each such solution must have the same elbow angle, as computed above.

Computation of feasible elbow points and wrist angles: To compute feasible elbow points, first compute the coordinates of the shoulder point with respect to the wrist point, expressed in the flange frame. This can be computed as: ${}^{flange}T_{shx}={}^{utorso}T_{shx}^{-1}\,{}^{utorso}T_{flange}$
Both terms on the right-hand side above have already been introduced. The 4th column of the result, ${}^{flange}T_{shx}$, expresses the shoulder point in flange coordinates: $s_{/flange}$.

Figure 2 from [1] is repeated here in Fig 1, illustrating constraints on the elbow point.

Since the wrist is fixed based on the target hand pose, the elbow point is constrained to lie in a plane perpendicular to the wrist axis at a radius $L_{forearm}$. Expressed in the flange frame, the elbow point must lie in the x-y plane on a circle with origin (0,0) and radius $L_{forearm}$. In addition, the elbow point must lie on the surface of a sphere, centered at the shoulder point with a radius of $L_{humerus}$.

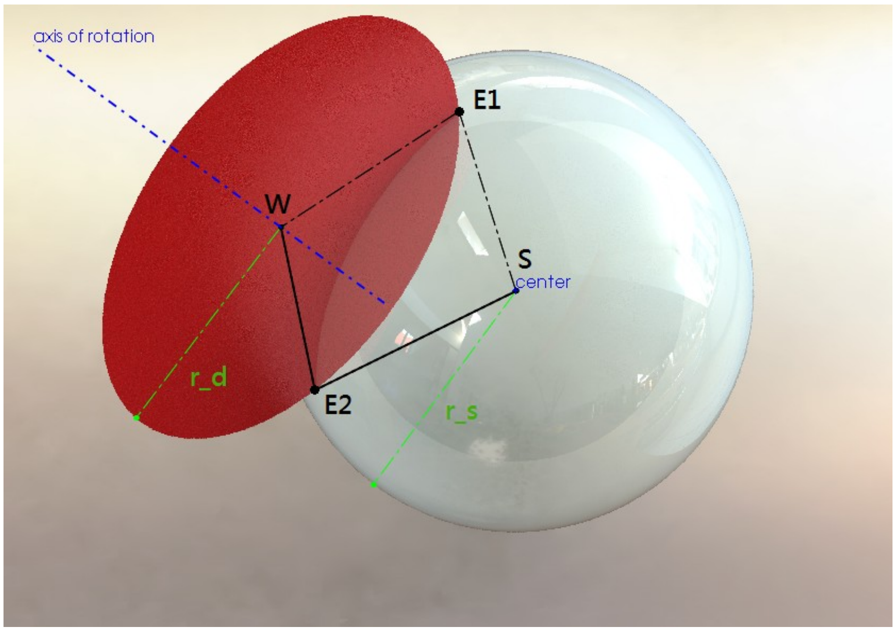

Illustration 1: Geometric constraints on elbow point from wrist disk and shoulder sphere

Viable elbow points occur at the intersection of the defined circle and sphere.

As a pre-condition, the x-y plane (in the flange frame) must intersect the shoulder sphere. This can be tested by examining the z-component of the shoulder point, as expressed in the flange frame. If $|s_{z/flange}| \geqslant L_{humerus}$, then the flange-frame x-y plane does *not* intersect the shoulder sphere. In this case, the algorithm should return that no solutions exist.

Assuming $|s_{z/flange}| < L_{humerus}$, the algorithm may continue—although a solution is not yet assured. The intersection between the flange-frame x-y plane and the shoulder sphere is a circle. This circle's center lies in the flange-frame x-y plane at $\boldsymbol{c}_{sliceCirlce/flange} = (s_x, s_y)_{/flange}$, and the radius of this circle, $r_{sliceCircle}$, must satisfy $r^2_{sliceCircle} = L^2_{humerus} - s^2_z$. If $\|\boldsymbol{c}_{sliceCircle}\|^2 > L^2_{forearm} + r^2_{sliceCircle}$, then the goal is unreachable, and the algorithm should return that no solutions exist.

If $\|\boldsymbol{c}_{sliceCircle}\|^2 < L^2_{forearm} + r^2_{sliceCircle}$, then the algorithm may progress to compute the possible elbow points. (The equality condition implies the goal is barely reachable; while a mathematical solution would exist, it would not be practical to use, due to its sensitivity). If this inequality is satisfied, then the slice circle and the wrist circle intersect in 2 points. These points can be computed by considering triangles of **w**, **s** and **e** with sides $L_{forearm}$, $r_{sliceCircle}$ and $\|\boldsymbol{c}_{sliceCircle}\|$. These two solutions for **e** are symmetric about the vector $\boldsymbol{c}_{sliceCirlce}$ at +/- wrist angles , Δq_w, which follows from a law of cosines: $r^2_{circleSlice} = L^2_{forearm} + \|\boldsymbol{c}_{sliceCircle}\|^2 - 2 L_{forearm} \|\boldsymbol{c}_{sliceCircle}\| \cos(\Delta q_w)$. The resultant solutions for wrist bend are $q_w = -atan2(\boldsymbol{c}_y, \boldsymbol{c}_x) \pm \Delta q_w$. The coordinates of the elbow point, as expressed in the flange frame, follow from: $\boldsymbol{e}_{/flange} = -L_{forearm}[\cos(q_w); \sin(q_w); 0]$.

The two solutions for **e** are $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, corresponding to the two elbow-angle solutions. The elbow angle solutions should be checked against the physical range limits. This may result in 0 viable solutions (at which point the algorithm terminates), or 1 or 2 viable solutions.

Computation of usy and shx angles: For the one or two candidate elbow points, these can be expressed in shx-frame coordinates with: $\boldsymbol{e}_{/usy} = {}^{utorso}T^{-1}_{usy} \, {}^{utorso}T_{flange} \boldsymbol{e}_{/flange}$. Given numerical values for $\boldsymbol{e}_{/flange}$, we can compute corresponding numerical values for $\boldsymbol{e}_{/usy}$.

We can also derive a second expression for $\boldsymbol{e}_{/usy}$, expressed abstractly in terms of the first two joint angles, q_{usy} and q_{shx}. This second expression is derived as follows.

The elbow point, expressed in utorso coordinates, must match the forward kinematics from utorso to the elbow (elx) frame. This can be obtained from the transform: ${}^{usy}T_{elx} = {}^{usy}T_{shx} \, {}^{shx}T_{ely} \, {}^{ely}T_{elx}$

From our earlier definition of ${}^{usy}T_{shx}$, and invoking the approximation $a_1 \approx 0$

$$ {}^{usy}T_{shx} \approx \begin{bmatrix} \cos(q_{usy}) & -\sin(q_{usy}) & \sin(q_{usy}) & 0.0 \\ \sin(q_{usy}) & 0 & -\cos(q_{usy}) & 0.0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Also, approximating $a_2 \approx 0$, we get:

$$ {}^{shx}T_{ely} \approx \begin{bmatrix} \cos(q_{shx}) & -\sin(q_{shx}) & \sin(q_{shx}) & 0.0 \\ \sin(q_{shx}) & 0 & -\cos(q_{shx}) & 0.0 \\ 0 & 1 & 0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Finally, approximating $a_3 \approx 0$, we get:

$$ {}^{ely}T_{elx} \approx \begin{bmatrix} \cos(q_{ely}) & -\sin(q_{ely}) & \sin(q_{ely}) & 0.0 \\ \sin(q_{ely}) & 0 & -\cos(q_{ely}) & 0.0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Multiplying these three transforms together, we get:

$$ {}^{usy}T_{elx} = {}^{usy}T_{shx}\,{}^{shx}T_{ely}\,{}^{ely}T_{elx} \approx \begin{bmatrix} \dots & \dots & \dots & \left(\cos(q_{usy})\sin(q_{shx}) + \sin(q_{usy})\cos(q_{shx})\right) d_3 \\ \dots & \dots & \dots & \sin(q_{usy})\sin(q_{shx}) d_3 \\ \dots & \dots & \dots & -\cos(q_{shx}) d_3 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The terms in the upper-left 3x3 orientation matrix of the above transform are not shown, since they are not needed (yet). The fourth column, however, (rows 1 through 3) provide an expression for $\boldsymbol{e}_{/usy}$. Note that, although we included the transform that is a function of q_{ely} , this angle does not appear in the fourth column. This can be visualized: rotation of the humerus does not affect the coordinates of the elbow (although q_{ely} does affect the orientation of the elbow frame).

Since our target hand pose and circle/sphere logic has led us to numerical expressions for $\boldsymbol{e}_{/usy}$, we can solve for angles q_{usy} and q_{shx} . To do so, start with setting element (3,4) in the above approximate transform ${}^{usy}T_{elx}$ to the known quantity $e_{z/usy}$. Using an arc-cosine, there are two possible solutions for q_{shx} . However, considering physical joint-range limitations, typically only one of these two possibilities (for given elbow point) is permissible for Atlas.

Continuing, given the elbow point and given the solution(s) q_{shx} , we can equate element (1,4) of ${}^{usy}T_{elx}$ to known quantity , $e_{x/usy}$ and (substituting in a solution for q_{shx}) q_{usy} follows from an arc-tan.

Considering two options for $\boldsymbol{e}_{/usy}$, possibly two options for q_{shx} (for each elbow point) and two options for q_{usy} for each of these cases, there are up to 8 possible solutions of (q_{usy}, q_{shx}) . However, in practice, the physical joint range limits conspire to restrict viable solutions to at most one solution (q_{usy}, q_{shx}) per elbow point candidate $\boldsymbol{e}_{/usy}$.

A singularity does exist for the special case of **w**, **s** and **e** all colinear and all lying along the usy joint axis (i.e. arm fully extended parallel to the usy joint). This special case needs to be treated separately—although it can be avoided if solutions of fully-outstretched arms are avoided.

Solution for forearm and humerus rotations:
If, at this point, there are viable solutions of $(q_{usy}, q_{shx}, q_{mwx}, q_{elx})$ (two shoulder joints, the elbow bend and the wrist bend), then the remaining two angles, (q_{ely}, q_{uwy}) , corresponding to the humerus rotation and the forearm rotation, need to be investigated. To do so, we may compute the axis direction of the elbow joint, which is the normal of the triangle with vertices **s**, **e**, **w**. The

directions of the wrist-bend and shoulder-shx joint axes are also known from the solution so far, $(q_{usy}, q_{shx}, q_{mwx}, q_{elx})$. It then follows that q_{uwy} (forearm rotation) is the rotation from elbow-axis to wrist-axis about the vector from **e** to **w**. Similarly, q_{ely} (humerus rotation) is the rotation from shoulder-shx axis to elbow-axis about the vector from **s** to **e**. In each case, there are two solutions (one positive and one negative). However, at most one of these solutions will lie within the joint limits.

Having computed q_{uwy} and q_{ely} , if there are viable solutions for both of these (and viable solutions for the other 4 joints), then a complete kinematic solution exists for all six joints. In fact, there may be two viable solutions (corresponding to the two elbow-point candidates). Empirically, if one inverse-kinematic solution exists, a second kinematic solution also exists approximately 30% of the time.

Precise solutions from approximate solutions:
If an approximate joint-space solution exists, it will have the properties that the forward-kinematic solution resulting from this approximate inverse-kinematic solution will match the desired hand orientation exactly. However, the location of the hand-frame origin will be off by about 2 cm (on average). The approximate solution can be used to try to find a precise inverse-kinematic solution. This is performed as follows:

1) initialize q_est to the approximate analytic solution;
2) compute forward kinematics to get **p** and **R** from $\boldsymbol{q}_{est}$, as well as the Jacobian, **J** (6x6, in this case)
3) compute the hand error in terms of $\Delta \boldsymbol{p} = \boldsymbol{p}_{des} - \boldsymbol{p}(\boldsymbol{q}_{est})$ and $\Delta \boldsymbol{\varphi} = q_{rot} \hat{\boldsymbol{a}}$, where $(q_{rot}, \hat{\boldsymbol{a}})$ = (angle,axis) of $\boldsymbol{R}_{des} \boldsymbol{R}^{-1}(\boldsymbol{q}_{est})$
4) construct a nominal pose error as $\Delta \boldsymbol{t} = [\Delta \boldsymbol{p}; \Delta \boldsymbol{\varphi}]$ (a 6x1 column vector).
5) If $||\Delta \boldsymbol{t}|| < \delta_{tolerance}$, ***halt***. Return $\boldsymbol{q}_{est}$ as the solution.
6) Otherwise, compute $\Delta \boldsymbol{q} = \boldsymbol{J}^{-1} \Delta \boldsymbol{t}$
7) Scale $\Delta \boldsymbol{q}$. If $||\Delta \boldsymbol{q}|| > \epsilon$, $\delta \boldsymbol{q} \equiv \epsilon \Delta \boldsymbol{q} / ||\Delta \boldsymbol{q}||$, else $\delta \boldsymbol{q} \equiv \Delta \boldsymbol{q}$
8) Update $\boldsymbol{q}_{est}$ as $\boldsymbol{q}_{est} += \delta \boldsymbol{q}$
9) If reached max number of iterations, return “no solution found”
10) loop back to step (2)

There is not a complete correspondence between approximate and precise solutions. A precise solution may exist even though an approximate solution does not exist. However, such cases are likely to be poor operating poses. Additionally, an approximate solution may exist, but a precise solution does not. Further, multiple (2) approximate solutions may exists, whereas the number of precise solutions may be 0, 1 or 2.

The HKU Inverse Kinematics Package for Atlas's Arms:
The HKU package “atlas_hand_kinematics” creates a library of kinematics functions that may be included in user programs. An example of how to use this package is the program “example_atlas_inverse_kinematics.cpp” in the package “example_ik.” Note that the package.xml file must include reference to “atlas_hand_kinematics” in order to include this library.

The atlas_hand_kinematics library implements the inverse-kinematics equations described herein. In “example_atlas_inverse_kinematics.cpp”, the line:

```
    ik_solver_ = new atlas_hand_ik_solver("rhand", "/utorso");
```

instantiates an object with various kinematics functions. It is initialized, in this example, to use the right hand and to use “utorso” as the base frame. That is, target hand poses should be expressed with respect to the chest (utorso) frame.

The callback function “markerFeedback” is set up to listen to messages from the example interactive-marker code (in package: example_interactive_marker). Incoming frames are re-interpreted in terms of origin (with respect to named frame) and orientation matrix (a 3x3 rotation matrix). These are used to populate an object of type Eigen::Affine3d, which is equivalent to a 4x4 transform matrix, but does not bother to store the fourth row (which is always [0,0,0,1]).

The affine transform, a_hand_desired_, is global in this example, so it is accessible by other functions within the same program.

A second callback function, “clickCB”, waits for an incoming message of type std_msgs::Bool on topic “ik_click.” The intent of this function is to have it perform its operations when it receives a button click. Implementation of a GUI to perform this function will be discussed in a separate document. In the mean time, this button-click message can be emulated manually by typing into at terminal:

```
 rostopic pub --once ik_click std_msgs/Bool 1
```

Sending this message will cause the “clickCB” callback function to attempt to find an inverse-kinematic solution to the most recent pose published by the interactive marker. Inverse-kinematic solutions are computed with the operation:

```
  ik_soln = ik_solver_->get_precise_solutions(a_hand_desired_);
```

The result of this function call can be that one or two valid IK solutions exist, or that no (precise) solutions were found. The method used is as described above, seeding a numerical search with an approximate solution from the analytical inverse kinematics.

If one or two valid solutions are found, they are printed to the terminal. Additionally, the corresponding marker origin is republished to the topic “marker_listener.” If desired, the marker publisher broker may be run as well, thus leaving a record (in rviz) that the frame considered did result in a successful inverse-kinematics solution. This feature may be useful for building up displays of reachability.

To run the example program, first start up gazebo/drcsim. Then run (in separate terminals):

```
 rosrun example_ik example_atlas_inverse_kinematics
 rosrun example_interactive_marker IM_example
```

As the interactive marker is moved, you will see a display in both of the above terminals of the corresponding coordinates of the interactive marker.

In a third terminal, enter the click emulation,

```
 rostopic pub --once ik_click std_msgs/Bool 1
```

and the example IK program will attempt to compute an IK solution and display the result.

A natural extension to this example is to include a third callback function that responds to a separate “click”, and which causes Atlas's arm to move to the most recently computed inverse-kinematic solution. This can be achieved by combining the present example code with the example code in the example_traj_client package.

Extensions:

The above description provides a means to discover precise inverse-kinematic solutions (if such exist) given a fully-specified (6-DOF) desired hand pose. For Atlas, it is actually quite rare that an inverse-kinematic solution exists for a given desired hand pose. However, sometimes a specific hand pose is not required, but a range of hand poses may be adequate, which increases the chances for finding a viable solution. For example to grasp a sphere sitting on a table, a good strategy is to approach the sphere from above, with the palm origin centered on the sphere (in x-y) and with the palm normal pointing downwards. In this case, there is one unconstrained (unspecified) degree of freedom of the desired hand pose. Equivalently, the hand-frame x-axis may be at any angle in the (x-y) plane (i.e., any direction that is perpendicular to the palm normal). The y-axis follows from the choice of x-axis, so there is only one degree of freedom unconstrained.

An approach to considering this case is to sample a set of x-axis directions (lying in the utorso x-y plane), and attempt to find precise inverse-kinematics solutions for these hand-pose options. If any such solution is found, this solution may be used to approach and grasp the sphere. Additional cases, including turning a valve and turning a door handle are described in [1].

An anticipated extension is computation of solutions with kinematic redundancy, if/when a seventh axis is added to Atlas' arms. In this case, a possible approximate solution may involve a sphere about the wrist point (of radius equal to the forearm length) and a sphere about the shoulder (of radius equal to the humerus length). If these two spheres intersect (given a desired hand pose), then there is a continuum of candidate elbow points at the (circular) intersection of the two spheres. These candidate elbow points may be sampled and solved separately using the existing 6-DOF algorithm logic.

References:

[1] Cong Du and Wyatt Newman, “Manipulation Planning for the Atlas Humanoid Robot”, to appear in ROBIO 2014.