ASBR THA2 Functions Documentation

Daniyal Maroufi and Anas Yousaf

adjointMatrix.m

Description: Calculates the adjoint representation of a transformation matrix.

Inputs: - `T`: Transformation matrix.

Outputs: - `Adjoint_T`: Adjoint representation of the transformation matrix.

transfmat2twist.m

Description: Converts the transformation matrix of a transformation in a 3D space into its corresponding twist vector representation.

Inputs: - `transfmat`: 4x4 transformation matrix.

Outputs: - `twist`: 6-element twist vector.

transfmat2screw.m

Description: Converts the transformation matrix of a transformation in a 3D space into its corresponding screw axis representation.

Inputs: - `transfmat`: 4x4 transformation matrix.

Outputs: - `point`: 3-element vector describing the point on the rotation axis. - `axis`: 3-element vector describing the rotation axis. - `pitch`: Scalar element describing the pitch of the screw axis. - `angle`: Scalar angle in radians describing the distance along the screw.

skew_to_screw_axis.m

Description: Converts a 4x4 skew symmetric matrix to a 6x1 screw axis.

Inputs: - `M`: 4x4 skew symmetric matrix.

Outputs: - `V`: 6x1 vector.

screw2twist.m

Description: Converts a screw axis representation to a twist vector.

Inputs: - `point`: 3-element vector describing the point on the rotation axis. - `axis`: 3-element vector describing the rotation axis. - `pitch`: Scalar element describing the pitch of the screw axis.

Outputs: - `twist`: 6-element twist vector.

rotmat2axisangle.m

Description: Converts a rotation matrix into its corresponding axis-angle representation.

Inputs: - `R`: 3x3 rotation matrix.

Outputs: - `axis`: 3-element column vector representing the axis of rotation. - `angle`: Scalar value representing the angle of rotation in radians.

is_valid_transform_matrix.m

Description: Determines if an input matrix is a valid 4x4 homogeneous transformation.

Inputs: - `transfmat`: 4x4 transformation matrix.

Outputs: - `result`: Returns true if valid matrix, false otherwise.

is_valid_rotation_matrix.m

Description: Checks if the input matrix is a valid rotation matrix.

Inputs: - `R`: 3x3 matrix to be checked. - `check_identity` (optional): Boolean flag to check if the matrix is the identity matrix (default: true).

Outputs: - `is_valid`: Boolean indicating whether the input matrix is a valid rotation matrix.

isequal_tol.m

Description: Checks if two matrices are equal within a specified tolerance.

Inputs: - `A`: First matrix. - `B`: Second matrix. - `tol` (optional): Tolerance value (default: 1e-6).

Outputs: - `isEqual`: Boolean indicating whether the matrices are equal within the specified tolerance.

J_space.m

Description: Calculates the space Jacobian of a robot using screw theory.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `q`: Joint configuration.

Outputs: - `Js`: Space Jacobian matrix.

J_body.m

Description: Calculates the body Jacobian of a robot using screw theory.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `q`: Joint configuration.

Outputs: - `Jb`: Body Jacobian matrix.

FK_space.m

Description: Calculates the forward kinematics of a robot using screw theory in the space frame.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `q`: Joint configuration. - `show_plot` (optional): Boolean flag to show the plot of the robot configuration. - `new_fig` (optional): Boolean flag to create a new figure for the plot.

Outputs: - `T`: End-effector transformation matrix.

FK_body.m

Description: Calculates the forward kinematics of a robot using screw theory in the body frame.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `q`: Joint configuration. - `show_plot` (optional): Boolean flag to show the plot of the robot configuration. - `new_fig` (optional): Boolean flag to create a new figure for the plot.

Outputs: - `T`: End-effector transformation matrix.

singularity.m

Description: Determines whether a robot configuration is at or near a singularity by calculating the rank of the input Jacobian.

Inputs: - `J`: Jacobian matrix.

Outputs: - `singular`: Boolean indicating whether the manipulator is at singularity.

manipulabilityEllipsoid.m

Description: Calculates the manipulability ellipsoid for a given input Jacobian matrix.

Inputs: - `J`: m x n Jacobian matrix.

Outputs: - `directions`: m x m matrix with each column representing a unit vector corresponding to a principal semi-axis of the ellipsoid. - `lengths`: 1 x m vector containing the lengths of the m principal semi-axes.

J_isotropy.m

Description: Calculates the isotropy measure of a manipulability ellipsoid given an input Jacobian matrix.

Inputs: - `J`: Jacobian matrix.

Outputs: - `isotropy`: Isotropy measure of manipulability ellipsoid.

J_condition.m

Description: Calculates the condition number of a manipulability ellipsoid given an input Jacobian matrix.

Inputs: - `J`: Jacobian matrix.

Outputs: - `condition`: Condition number of manipulability ellipsoid.

J_ellipsoid_volume.m

Description: Calculates the ellipsoid volume of a manipulability ellipsoid given an input Jacobian matrix.

Inputs: - `J`: Jacobian matrix.

Outputs: - `volume`: Ellipsoid volume of manipulability ellipsoid.

plotTransformAxes.m

Description: Plots the axes of a frame defined by the input transformation matrix with the specified color and label.

Inputs: - `transfmat`: 4x4 transformation matrix. - `color`: Color of the axes. - `label`: Label of the axes.

Outputs: - `axes`: Column vector for each axis that is plotted.

plotScrewAxis.m

Description: Plots the screw axis defined by the input in a 3D space.

Inputs: - `point`: 3-element vector describing the point on the rotation axis. - `axis`: 3-element vector describing the rotation axis. - `pitch`: Scalar element describing the pitch of the screw axis.

plotEllipsoid.m

Description: Plots a 3D ellipsoid given an input center location vector and principal semi-axes vectors.

Inputs: - `center`: 3-element vector containing the center of the ellipsoid. - `axes`: Matrix with each column defining a principal semi-axis of the ellipsoid.

ellipsoid_plot_linear.m

Description: Calculates and plots the linear velocity manipulability ellipsoid of a robot configuration.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `q`: Joint configuration.

ellipsoid_plot_angular.m

Description: Calculates and plots the angular velocity manipulability ellipsoid of a robot configuration.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `q`: Joint configuration.

redundancy_resolution.m

Description: Implements the redundancy resolution algorithm for a redundant robot using the manipulability measure.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `Ti`: Initial end-effector transformation matrix. - `Tf`: Desired end-effector transformation matrix. - `q0`: Initial joint configuration. - `max_iterations`: Maximum number of iterations. - `K`: Gain for the manipulability measure.

Outputs: - `q`: Joint configuration. - `e`: Error vector.

J_inverse_kinematics.m

Description: Calculates the inverse kinematics of a robot using the Jacobian.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `Ti`: Initial end-effector transformation matrix. - `Tf`: Desired end-effector transformation matrix. - `q0`: Initial joint configuration. - `max_iterations`: Maximum number of iterations.

Outputs: - `q`: Joint configuration. - `e`: Error vector.

J_transpose_kinematics.m

Description: Solves inverse kinematics using the Jacobian transpose method.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `q_init`: Initial joint configuration. - `T_desired`: Desired end-effector transformation matrix.

Outputs: - `q_sol`: Final joint configuration. - `errors`: Error vector. - `joint_configs`: Joint configurations during iterations.

`DLS_inverse_kinematics`

Calculates the inverse kinematics of a robot using the Damped Least Squares (DLS) method.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `Ti` - initial end-effector transformation matrix - `Tf` - desired end-effector transformation matrix - `q0` - initial joint configuration - `max_iterations` - maximum number of iterations - `lambda` - damping factor for the DLS method

Outputs: - `q` - joint configuration - `e` - error vector

redundancy_resolution.m

This function implements the redundancy resolution algorithm for a redundant robot. The algorithm uses the manipulability measure to resolve the redundancy.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `Ti` - initial end-effector transformation matrix - `Tf` - desired end-effector transformation matrix - `q0` - initial joint configuration - `max_iterations` - maximum number of iterations - `K` - Gain for the manipulability measure.

Outputs: - `q` - joint configuration - `e` - error vector

`DLS_inverse_kinematics`

Calculates the inverse kinematics of a robot using the Damped Least Squares (DLS) method.

Inputs: - `robot`: Struct containing robot kinematics information. - `n_joints`: Number of joints in the robot. - `M`: 4x4 transformation matrix representing the end-effector pose in the home (zero) configuration. - `space.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the space frame. - `body.screw_axes`: 6xN matrix where each column represents the screw axis of a joint in the body frame. - `joint_as_end_effector` (optional): 4x4xN transformation matrices representing the pose of each joint as the end-effector (used for plotting or visualization). - `Ti` - initial end-effector transformation matrix - `Tf` - desired end-effector transformation matrix - `q0` - initial joint configuration - `max_iterations` - maximum number of iterations - `lambda` - damping factor for the DLS method

Outputs: - `q` - joint configuration - `e` - error vector