The ScorBot Toolbox for MATLAB: An Open-Source Hardware Interaction and Simulation Library for the Intelitek SCORBOT-ER 4u Educational Robot

Prof. Michael Dennis Mays Kutzer, United States Naval Academy

Michael D. M. Kutzer received his Ph.D. in mechanical engineering from the Johns Hopkins University, Baltimore, MD, USA in 2012. He is currently an Associate Professor in the Weapons, Robotics, and Control Engineering Department (WRCE) at the United States Naval Academy (USNA). Prior to joining USNA, he worked as a senior researcher in the Research and Exploratory Development Department of the Johns Hopkins University Applied Physics Laboratory (JHU/APL). His research interests include robotic manipulation, computer vision and motion capture, applications of and extensions to additive manufacturing, mechanism design and characterization, continuum manipulators, redundant mechanisms, and modular systems.

Dr. John S Donnal
Dr. Carl E. Wick Sr., United States Naval Academy

Dr. Carl Wick is currently a Professional Lecturer with the Biomedical Engineering Department of the George Washington University where he provides technical assistance and advice to capstone project students. Previously he was associated with the U.S. Na

The ScorBot Toolbox for MATLAB: An Open-Source Hardware Interaction and Simulation Library for the Intelitek SCORBOT-ER 4u Educational Robot

Abstract

An open-source software tool developed for the Intelitek SCORBOT-ER 4u Educational Robot with a MATLAB front-end is presented. This “ScorBot Toolbox” provides a documented, user-friendly, and open-source tool for installation, hardware interaction, kinematic modeling, and visualization using MATLAB. This paper describes the motivation, development, features, and limitations of the ScorBot Toolbox; and illustrates its capabilities in the context of in-person and remote project-based learning (PBL). Source code, documentation, and installation functions for the ScorBot Toolbox are available at *https://github.com/kutzer/ScorBotToolbox*.

1 Introduction

The SCORBOT-ER 4u Educational Robot (Intelitek Inc, Derry, NH) is a five Degree-of-Freedom (DoF) articulated manipulator with an integrated 65 mm (2.6 in) stroke, electric gripper. The SCORBOT-ER 4u manipulator is a lightweight, 10.8 kg (23.8 lbs) system with a maximum linear speed of 700 mm/sec (27.6 in/sec). Though not a collaborative manipulator, this low-mass and low-speed system leverages a belt-drive design to create a low-inertia platform that is safe for educational use. Similar to its industrial counterparts, the SCORBOT-ER 4u system includes a manipulator, controller, teach pendant, and proprietary programming environment [1]. Figure 1 summarizes the SCORBOT-ER 4u system.

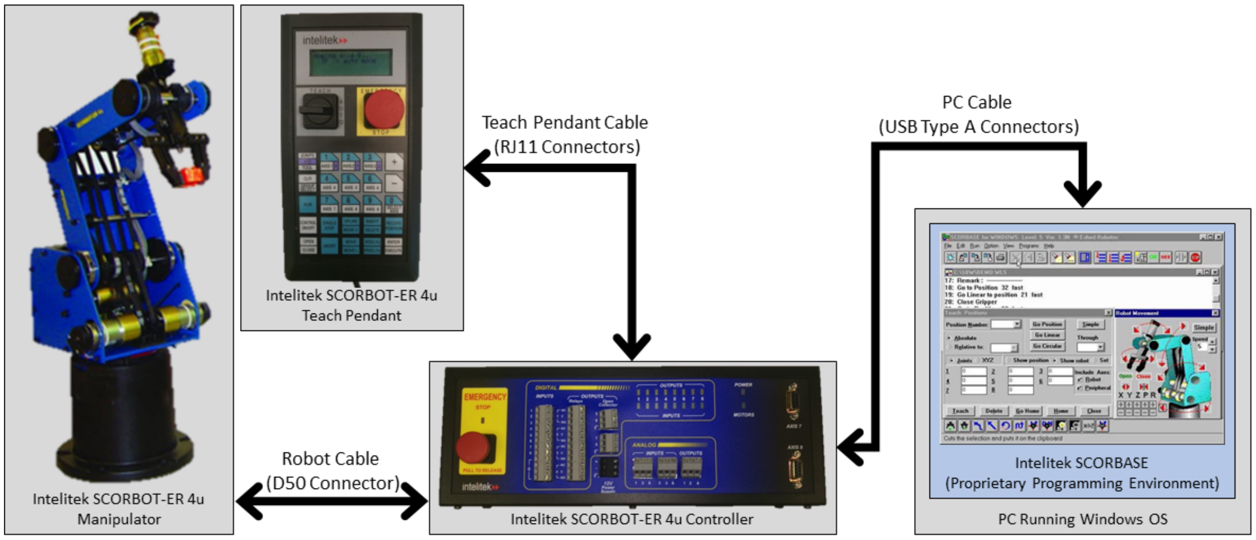
Intelitek SCORBOT-ER 4u Manipulator
Intelitek SCORBOT-ER 4u Teach Pendant
Teach Pendant Cable (RJ11 Connectors)
PC Cable (USB Type A Connectors)
Robot Cable (D50 Connector)
Intelitek SCORBOT-ER 4u Controller
Intelitek SCORBASE (Proprietary Programming Environment)
PC Running Windows OS

Figure 1: Summary of the SCORBOT-ER 4u Educational Robot system comprised of a manipulator, controller, teach pendant, and proprietary programming environment (SCORBASE).

In 2011, Esposito et al presented the “MATLAB Toolbox for the Intelitek Scorbot (MTIS)” [2] which provides basic interaction between a 32-bit version of MATLAB running on a Windows Operating System (OS) and the Intelitek SCORBOT-ER 4u. The motivation and inspiration for MTIS were driven by: (1) a desire to increase hands-on exercises shown to particularly benefit visual and experiential learners [3]; (2) the popularity of Intelitek SCORBOT manipulators in education during the 1990s and early 2000s; and (3) a desire to follow a then-recent trend of providing free, open-source software for robotic education and research.

As described by [2], MTIS successfully provides a MATLAB front-end for interacting with the USB version of the SCORBOT-ER 4u. MTIS’s functionality is based on a United States Naval Academy (USNA) developed “MTIS Intermediary DLL” allowing MATLAB to interact with an “Intelitek Name Mangled DLL.” For clarity, the “Intelitek Name Mangled DLL” refers to USBC.dll [4] which is included as part of the Intelitek SCORBASE software [1]. Beta testing of MTIS indicated that the tool was generally well-received by students in an introductory robotics course, and the only documented complaint related to moderate or frequent instances of the MTIS software crashing [2].

Inspired by the success of MTIS and over five years of interaction with MTIS in a classroom environment, the ScorBot Toolbox was developed incrementally with the following design goals: (1) Usability, (2) Consistency, and (3) Stability.

This paper describes the development and use of the ScorBot Toolbox. The ScorBot Toolbox provides a series of seamless, self-contained MATLAB functions allowing users to control a SCORBOT-ER 4u, receive feedback from a SCORBOT-ER 4u, model elements of the SCORBOT-ER 4u, and create 3D rendered visualizations of the SCORBOT-ER 4u.

This paper is organized as follows: Section 2 describes work to improve the usability of the ScorBot Toolbox through installation and version control tools, 64-bit support, and visualization tools for cross-platform compatibility; Section 3 describes the organization, naming, syntax, and visual debugging capabilities of the toolbox; Section 4 describes methods implemented to reduce and eliminate crashes, provide straightforward error handling and descriptions, and bug tracking tools implemented within the toolbox. Section 5 provides a comparison between MTIS and the ScorBot Toolbox, provides a comparison between hardware and simulation syntax, and describes an example of the PBL laboratory uses of the toolbox’s “digital twin” capabilities leveraged during COVID hybrid learning. Section 6 discusses the implications of the results presented in Section 5 and includes a summary of qualitative student feedback. Finally, Section 7 summarizes the conclusions that can be drawn from this work and discusses related future work.

2 Usability Considerations

This section describes efforts to improve the usability of the ScorBot Toolbox through installation and version control tools, 64-bit support, and visualization tools for cross-platform compatibility. Section 2.1 describes the installation and version control approach used; Section 2.2 describes the approach to providing 64-bit support; and Section 2.3 describes visualization tools developed to provide cross-platform compatibility.

2.1 Installation and Version Control

Installation, update, and version control capabilities are motivated by a desire to rapidly provide toolbox updates including capabilities expansions and bug fixes. The ScorBot Toolbox software is maintained using a Git repository hosted by GitHub, Inc. Hosting through GitHub provides both reliable version control and trusted integration with the MATLAB Central File Exchange. As a result, archived branches are accessible for download and extraction using the MATLAB “unzip” function.

First-time installation of the ScorBot Toolbox [5] requires several manual steps and loosely mimics the installation procedure of MTIS:

1. Download “ScorBotToolbox.zip” (or alternate filename)
2. Unzip “ScorBotToolbox.zip”
3. Open MATLAB as an administrator
4. Change your MATLAB *Current Directory* to the location containing contents of the unzipped ScorBotToolbox
5. Run `installScorBotToolbox`
6. (Optional) Manually move “ScorBotToolbox Example SCRIPTS” to a location of your choosing

Unlike MTIS, the ScorBot Toolbox includes automated installation and update functions. Automated installation, housed in the `installScorBotToolbox` function, executes the following steps:

1. Download, unzip and install required toolboxes and support packages [6–10]
2. Find and close open and/or running toolbox components
3. Find and remove old toolbox versions
4. If a 64-bit Windows OS is used
   (a) Install, configure, and authorize “ScorBotServer”
   (b) Copy toolbox *hardware-related* functions and support files to the toolbox root ([MATLAB Root], ’toolbox’, ’scorbot’).
   (c) Copy the latest version of USBC.dll from the SCORBASE directory to the ScorBotServer directory.
5. Copy toolbox *modeling and simulation-related* functions and support files to the toolbox root ([MATLAB Root], ’toolbox’, ’scorbot’).
6. Rehash the MATLAB toolbox cache.

Including this installation function in a known location within the toolbox enables a simple update function to download and unzip the archived toolbox branch (using the MATLAB “unzip” function), and execute the toolbox-specific install function. The resultant update procedure for the ScorBot Toolbox is as follows:

1. Open MATLAB as an administrator
2. Run ”ScorUpdate”

This reliable automation for installing and updating the ScorBot Toolbox provides a pipeline for rapid updates and bug fixes. Further, applying this install/update functionality to other toolboxes enables the “install required toolboxes and support packages” functionality. This secondary benefit allows generic functions to be maintained separately and used outside of the ScorBot Toolbox context. The use of Git provides version control and branching capabilities that simplify development, and the selection of GitHub provides streamlined interaction with existing MATLAB functionality.

2.2 64-bit Support

The requirement for 64-bit support is driven by the following: (1) the last release of MATLAB supported on a 32-bit Windows OS was R2015b; (2) MATLAB 64-bit versions cannot load 32-bit Windows OS DLLs; (3) Intelitek will only release a 32-bit Windows OS version of the USBC DLL; and (4) the MTIS Intermediary DLL must be 32-bit to interact with the USBC DLL.

Support for 64-bit, therefore, requires the introduction of an additional intermediary that: (1) supports simple MATLAB interaction; and (2) can access and interact with the 32-bit MTIS Intermediary DLL. This is accomplished with the introduction of the “ScorBotServer” executable. When run, ScorBotServer creates a local web server wrapper for the MTIS Intermediary DLL. This allows MATLAB to access MTIS Intermediary DLL functions using encoded URLs, and receive feedback from the MTIS Intermediary DLL through JSON messages posted to said encoded URLs. A summary of this can be found in Figure 2.

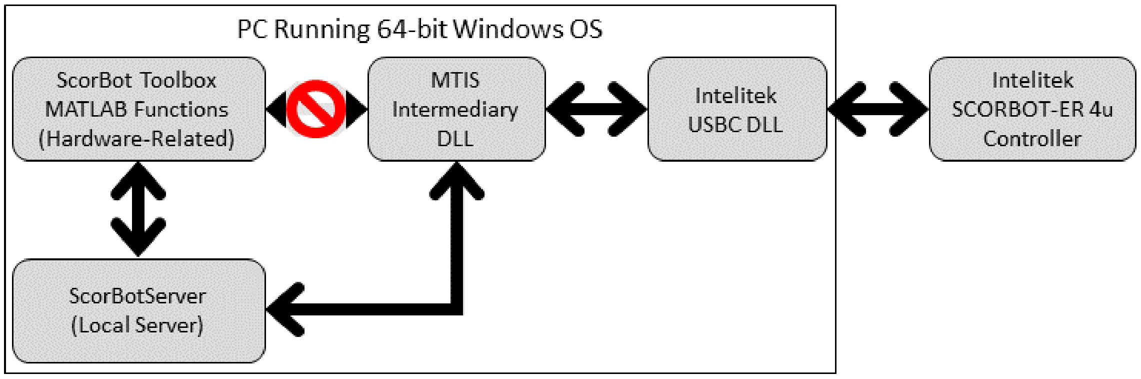
PC Running 64-bit Windows OS
ScorBot Toolbox MATLAB Functions (Hardware-Related)
MTIS Intermediary DLL
Intelitek USBC DLL
Intelitek SCORBOT-ER 4u Controller
ScorBotServer (Local Server)

Figure 2: The communication structure of the ScorBot Toolbox. Hardware-related toolbox functions communicate with the MTIS Intermediary DLL through the ScorBot Server with encoded URLs.

To ensure ease of use: (1) ScorBot server is installed and configured automatically during the `installScorBotToolbox` process; and (2) when interacting with hardware, the ScorBotServer is initialized automatically as a background application using the `system` function in MATLAB.

2.3 Cross-Platform Compatible Simulation Tools

The 32-bit Windows OS limitation imposed by the USBC DLL provided by Intelitek restricts hardware interaction with the SCORBOT-ER 4u to a Windows OS running natively on a PC or virtual machine (VM). To support non-Windows users without access to a Windows VM, the ScorBot Toolbox is built to include kinematic simulation and visualization tools that mimic hardware functionality. These tools allow users to interact with a kinematically equivalent model and high-fidelity 3D visualization of the SCORBOT-ER 4u rather than the physical hardware.

Leveraging timer-based callback functions in MATLAB, the ScorBot Toolbox simulation tools move the robot in the background, allowing a user to continue to interact with the MATLAB simulation as they would with the physical robot hardware. This provides the capability to write simulation programs that translate directly to programs written for hardware.

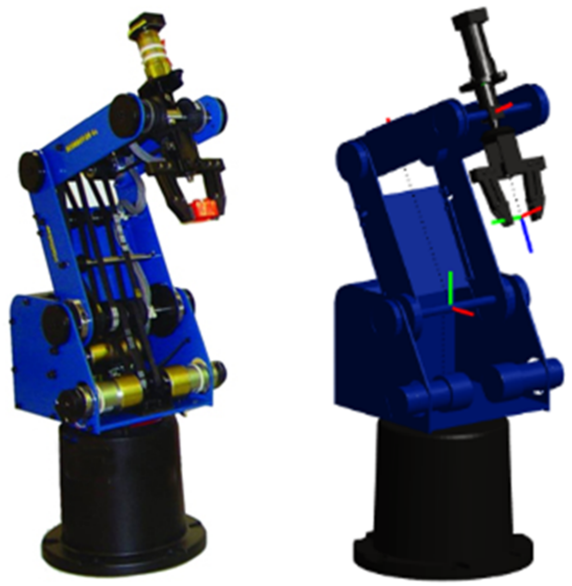

Figure 3: True SCORBOT-ER 4u hardware (left) compared to the 3D visualization included with the ScorBot Toolbox simulation (right).

3 Design for Consistency

The ScorBot Toolbox leverages a function naming convention inspired by MTIS. All user-level MATLAB functions within the toolbox begin with "Scor" followed by one or more modifier(s) that identify the specific toolbox function. Table 1 summarizes the 43 core functions included with the ScorBot Toolbox.

Table 1: Summary of key function names within the ScorBot Toolbox. For non-empty cells, columns labeled ***Modifier**** indicate optional modifiers and columns labeled **Modifier** indicate required modifiers.

	Base Name	***Modifier****	**Modifier**	***Modifier****	**Modifier**	**Modifier**
1	Scor	Sim	Init			
2	Scor		Home			
3	Scor	Sim	GoHome			
4	Scor	Sim	WaitForMove			
5	Scor	Sim	IsMoving			
6	Scor	Sim	Get		BSEPR	
7	Scor	Sim	Get		XYZPR	
8	Scor	Sim	Get		Pose	
9	Scor	Sim	Get		Gripper	
10	Scor	Sim	Get		Speed	
11	Scor		Get		PendantMode	
12	Scor	Sim	Set	Delta	BSEPR	
13	Scor	Sim	Set	Delta	XYZPR	
14	Scor	Sim	Set	Delta	Pose	
15	Scor	Sim	Set		Gripper	
16	Scor	Sim	Set		Speed	
17	Scor		Set		PendantMode	
18	Scor		BSEPR		2	XYZPR
19	Scor		BSEPR		2	Pose
20	Scor		XYZPR		2	BSEPR
21	Scor		XYZPR		2	Pose
22	Scor		Pose		2	BSEPR
23	Scor		Pose		2	XYZPR

As an example, the Row 1 of Table 1 indicates that the functions `ScorInit` and `ScorSimInit` exist; Row 12 indicates that the functions `ScorSetBSEPR`, `ScorSimSetBSEPR`, `ScorSetDeltaBSEPR`, and `ScorSimSetDeltaBSEPR` exist; and Row 18 indicates that only `ScorBSEPR2XYZPR` exists. Table 2 provides descriptions of the modifiers, and each function contains detailed help documentation resembling the help documentation available with native MATLAB functions.

Table 2: Descriptions of optional and required modifiers used in ScorBot Toolbox key function naming.

Modifier	**Description**
Sim	Functions containing “Sim” interact with the simulation environment rather than hardware.
Delta	Functions containing “Delta” prompt a relative movement of the hardware or simulation rather than an absolute movement relative to the system’s base frame.
Init	Functions containing “Init” prompt an initialization procedure for the hardware or simulation.
Home	Functions containing “Home” prompt a homing procedure for the hardware.
GoHome	Functions containing “GoHome” prompt a movement to the home position for the hardware or simulation.
WaitForMove	Functions containing “WaitForMove” pause MATLAB execution until a movement of the hardware or simulation has finished.
IsMoving	Functions containing “IsMoving” return a logical value indicating whether or not the hardware or simulation is moving. A value of “true” is returned if movement is occurring when the function is called.
Get	Functions containing “Get” query information from the hardware or simulation.
Set	Functions containing “Set” send information related to or triggering an action by the hardware or simulation.
BSEPR	Functions containing “BSEPR” accept or return the 5-element joint configuration (in radians) of the hardware or simulation. This is defined as a 1×5 array $[b \text{ (rad)}, s \text{ (rad)}, e \text{ (rad)}, p \text{ (rad)}, r \text{ (rad)}]$
XYZPR	Functions containing “XYZPR” accept or return the 5-element task configuration of the hardware or simulation. This is defined as a 1×5 array $[x^0 \text{ (mm)}, y^0 \text{ (mm)}, z^0 \text{ (mm)}, p^0 \text{ (rad)}, r \text{ (rad)}]$
Pose	Functions containing “Pose” accept or return a 4×4 array element of SE(3) (the special Euclidean group in three dimensions) defining the pose (position and orientation) of the hardware or simulation end-effector relative to the system’s base frame.
Gripper	Functions containing “Gripper” accept or return a positive scalar value describing the gripper state of the hardware or simulation in millimeters. A value of 0 (mm) indicates that the gripper is closed, and a value of 70 (mm) indicates that the gripper is fully open. The “Set” variations of these functions also accept string arguments of “Open” and “Close” to fully open or close the hardware or simulation gripper.
PendantMode	Functions containing “PendantMode” query hardware or prompt the user to change the teach pendant setting of the hardware between teach and auto.
2	Functions containing “2” indicate a conversion between configuration or pose information. As an example “ScorBSEPR2XYZPR” will accept a 5-element joint configuration, and return a 5-element task configuration.

4 Addressing Errors, Bugs, and Crashes

This section describes efforts to address errors, bugs, and crashes in the ScorBot Toolbox. Section 4.1 describes methods to provide error descriptions and mitigation recommendations; Section 4.2 describes error logging tools created to identify and fix recurring bugs; and Section 4.3 describes efforts to identify and eliminate the causes of ScorBot Toolbox crashes.

4.1 Error Handling

The ScorBot Toolbox considers three key types of errors:

1. *Function Syntax Errors* - errors triggered by calling hardware or simulation functions with improper syntax.
2. *Function Input Out-of-Bounds Errors* - errors triggered by attempting to set hardware or simulation values, or convert values outside of the prescribed limits.
3. *Hardware Errors* - errors triggered by hardware and communicated via the USBC DLL.

Addressing *Function Syntax Errors* and *Function Input Out-of-Bounds Errors* is a straightforward exercise in understanding the proper function syntax and variable limits. As functions are created and documented, code to describe and throw these errors can be added before the function is deployed. In general, errors are thrown using the MATLAB `error` function. In practice, some errors were shifted to warnings based on severity and consequence, and descriptions were modified based on student and instructor feedback.

Addressing *Hardware Errors* requires interpretation of the integer values returned by the USBC DLL. The extensive comments within the "Error.h" header file [11, 12] included with the installation of the Intelitek SCORBASE software provide verbal descriptions of many of the errors returned by the hardware. The function `ScorParseErrorCode` is included with the ScorBot Toolbox to parse, provide verbal interpretation, and mitigation strategies for these hardware-related errors.

The introduction of `ScorParseErrorCode` provides the ancillary benefit of allowing specific hardware errors to be flagged as fatal. In the event of a fatal error, the user is notified that: (1) the hardware is in an unrecoverable state, (2) this type of fatal error can be avoided in the future using described strategies, and (3) a restart is required to continue working with the hardware. While surprisingly simple, identifying these errors eliminates crashes by ceasing communication with the USBC DLL and allows users to save work before restarting. This one realization alleviated much of the frustration associated with the ScorBot Toolbox.

Stopping all hardware interaction following a fatal error proved somewhat complex. This is caused because the ScorBot Toolbox is modeled after MTIS. As a result, it consists of a large number of individual functions as compared to creating one or more custom objects (e.g. objects associated with a hardware and/or simulation class). To compensate, the ScorBot Toolbox introduces a hidden figure that acts as a pegboard for sharing background information across functions (including fatal error codes) and ensuring proper shutdown procedures using delete callback functions.

4.2 Error Logging and Bug Identification

Despite the noteworthy improvements associated with the error handling described in Section 4.1, intermittent and seemingly unpredictable errors continued to impact the performance of the ScorBot Toolbox. To track and identify the source of these issues, a series of error tracking and logging functions were created and included with the toolbox. Much like error logging in a commercial product, these functions create and update an error log that is saved on the local machine.

Contained within a single text file, a new error log is created each time the SCORBOT is initialized. A line is added to the error log in the event of a hardware error or use of a "ScorSet" command to interact with hardware. Lines within the error log contain the following information:

1. A date and time stamp
2. A USBC DLL error code or 0 if no error is present
3. A movement or error identification flag identifying log entries prompted by:
    - Error state;
    - Absolute movements evolving linearly in joint space;
    - Absolute movements evolving linearly in task space;
    - Relative movements evolving linearly in joint space;
    - Relative movements evolving linearly in task space; or
    - Movements of the gripper.
4. Current joint configuration (if prompted by an error state or `ScorSetGripper` command) or desired joint configuration (if prompted by a "ScorSet*" command, excluding gripper commands).
5. Current gripper state (if prompted by an error state or a "ScorSet*" command, excluding gripper commands) or desired gripper state (if prompted by a `ScorSetGripper` command).

These logs provide sufficient information to re-create errors associated with hardware interaction and provide insight into the patterns used by students during individual laboratory exercises. In general, these error logs provided insight to improve error descriptors to avoid incorrect sequences of commands and identify additional fatal errors and sequences of errors resulting in a crash. Additionally, these logs provided evidence suggesting the need for basic functionality like the `ScorSetUndo` command that allows a user to move to a previous waypoint.

4.3 Addressing Crashes

The introduction and use of error logging described in Section 4.2 further reduced the instances of crashes in the ScorBot Toolbox, but it did not fully eliminate their occurrence. In some instances, the sequence of events recorded in an error log associated with a crash would not reproduce a crash. In other instances, the sequence of events would reproduce a crash at one laboratory station, but not another.

For these cases, two culprits were identified:

1. Loose connections between the SCORBOT-ER 4u Controller and Teach Pendant; and
2. Out-of-date and/or mismatched configuration and library files

The SCORBOT-ER 4u Controller and Teach Pendant are connected using RJ11 cables that degrade with use. Unplugging the Teach Pendant from the Controller, even momentarily, causes the USBC DLL to crash. When this happens while running MTIS, MATLAB will crash entirely. When this happens running ScorBot Toolbox on a 64-bit Windows OS, ScorServer will crash. In both instances, some or all of the software must be reinitialized.

To reduce and potentially eliminate reliance on the Teach Pendant, the ScorBot Toolbox includes `ScorSimTeachBSEPR` and `ScorSimTeachXYZPR` with guided interfaces that allow the user to interact with the simulation environment using the number pad of the keyboard. Screenshots of the user interface for `ScorSimTeachBSEPR` and `ScorSimTeachXYZPR` is shown in Figure 4. These simulated movements can then be applied to hardware using combinations of the "ScorSimGet*" and "ScorSet*" commands.

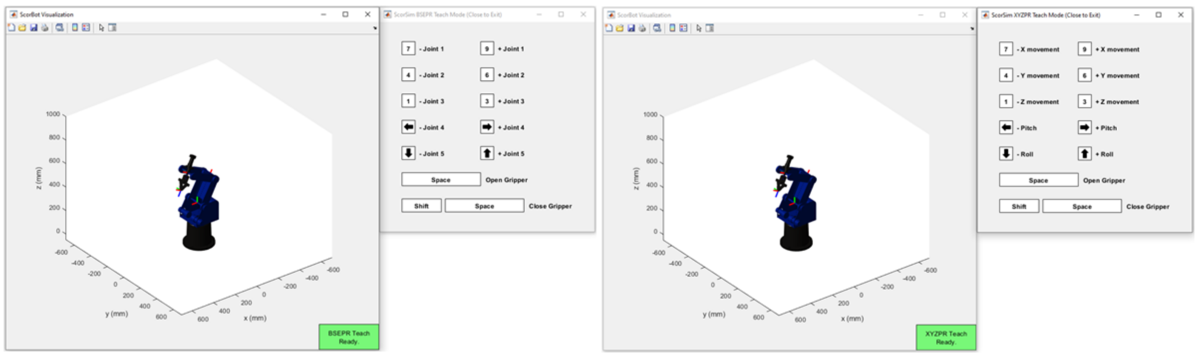

Figure 4: Guided interface allowing the user to interact with the ScorBot Toolbox simulation environment in joint space ("BSEPR", left), and in task space ("XYZPR", right).

Crashes apparently due to out-of-date and/or mismatched configuration and library files are highly unpredictable and notably frequent. Before the fix described below, these crashes occur if and when SCORBASE is upgraded after the ScorBot Toolbox was installed. Upgrading SCORBASE causes a mismatch between the configuration files and USBC library files used by the ScorBot Toolbox and SCORBASE. It is still unknown whether crashes were directly caused by this mismatch, but synchronizing the configuration and library files does fix the issue.

The function `ScorConfigurationSync` is included with the toolbox to ensure these configuration and library files are the same. Additionally, the automatic use of `ScorConfigurationSync` during installation and updates reduces the likelihood of mismatched files.

5 Results

For preliminary comparison, the capabilities of MTIS are compared to the ScorBot Toolbox. Assuming the utility of all content, capabilities are quantified in terms of the number of unique MATLAB functions, unique internal MATLAB functions (if applicable), unique MATLAB example scripts, total lines of actionable code, and total lines of comments. A summary of this comparison can be found in Table 3.

Table 3: Comparison between MTIS and the ScorBot Toolbox in terms of measurable attributes to the code. Note that the ScorBot Toolbox leverages code from five support toolboxes/packages that are automatically installed during installation and updates.

	MTIS [13]	**ScorBot Toolbox** [5]	**Support Toolboxes** [6–10]
Unique functions	34	128	243
Internal functions	0	12	41
Unique scripts	2	16	24
Lines of actionable code	846	18,986	33,752
Lines of comments	636	10,216	16,082

The contents of Table 3 attempt to quantify the difference in capabilities between MTIS and the ScorBot Toolbox. From Table 3, the total number of “unique functions” installed with MTIS is 34; and the total number of “unique functions” installed with the ScorBot Toolbox is 372 (128 ScorBot-specific functions and 243 support functions). In this context “unique functions” are defined as individual “*.m” files following the MATLAB function syntax. An “internal function” is defined in-line, within a “unique function,” and accessed only by the “unique function” containing it. “Unique scripts” are individual “*.m” files not containing the MATLAB function syntax. These “unique scripts” typically represent examples for new users.

As an alternative comparison, the final two rows of Table 3 compare the total number of actionable code and comment lines contained within MTIS and the ScorBot Toolbox. In this comparison, a “line of actionable code” is defined as a string containing valid MATLAB syntax separated by a linefeed, colon, or comma not preceded by an ellipsis or percent symbol. A “line of comments” is any string beginning with at least one percent symbol and ending with a linefeed. Lines contained between block-comment symbols are ignored in this analysis. From Table 3, the total number of “lines of actionable code” installed with MTIS is 846; and the total number of “lines of actionable code” installed with the ScorBot Toolbox is 52,738 (18,986 ScorBot-specific lines of code and 33,752 lines of support code). The number of “lines of comments” included with MTIS is 636 and 26,292 for the ScorBot Toolbox (10,216 ScorBot-specific comments, and 16,082 support code comments).

A general example of the digital twin capabilities included with the ScorBot Toolbox is highlighted in a comparison between Algorithm 1 and Algorithm 2. Each algorithm demonstrates initialization, basic movements, data acquisition, and gripper commands. Algorithm 1 provides this example using the simulation environment; and Algorithm 2 provides this example using commands controlling the SCORBOT-ER 4u hardware. Comparing lines between Algorithm 1 and Algorithm 2 shows the similarity in syntax between the simulation and hardware tools.

Algorithm 1: ScorBot Toolbox *simulation* example demonstrating: (1) initialization of the SCORBOT-ER 4u simulation; (2) a linear movement in task space; (3) a standard “wait for move” (pausing MATLAB’s execution until the movement completes); (4) a linear movement in joint space, data acquisition during a movement; (5) basic gripper commands; (6) and a command returning the SCORBOT-ER 4u simulation to the home configuration.

```
% Initialize ScorBot Toolbox Simulation
sim = ScorSimInit;
% Render 3D simulation geometry
ScorSimPatch(sim);

% Define desired waypoints as end-point XYZPR positions/orientations
XYZPR_i(1,:) = [500,-200,570,0,-2*pi/2];
XYZPR_i(2,:) = [500, 200,270,0, 0*pi/2];

% Convert waypoints to BSEPR
BSEPR_i(1,:) = ScorXYZPR2BSEPR( XYZPR_i(1,:) );
BSEPR_i(2,:) = ScorXYZPR2BSEPR( XYZPR_i(2,:) );

% Explore movement: Linear task movement with joint space waypoint
ScorSimSetBSEPR(sim,BSEPR_i(1,:),'MoveType','LinearTask');
ScorSimWaitForMove(sim);
% Explore movement: Linear joint movement with task space waypoint
ScorSimSetXYZPR(sim,XYZPR_i(2,:),'MoveType','LinearJoint');
% Collect movement data
XYZPRs = [];
BSEPRs = [];
while ScorSimIsMoving(sim)
    XYZPRs(end+1,:) = ScorSimGetXYZPR(sim);
    BSEPRs(end+1,:) = ScorSimGetBSEPR(sim);
end
% Explore movement: Open the gripper using a character array
ScorSimSetGripper(sim,'Open');
ScorSimWaitForMove(sim);
% Explore movement: Close the gripper using a scalar value
ScorSimSetGripper(sim,0);
ScorSimWaitForMove(sim);
% Return simulation to the home position
ScorSimGoHome(sim);
ScorSimWaitForMove(sim);
```

Algorithm 2: ScorBot Toolbox *hardware* example demonstrating: (1) initialization and homing of the SCORBOT-ER 4u hardware; (2) a linear movement in task space; (3) a standard “wait for move” (pausing MATLAB’s execution until the movement completes); (4) a linear movement in joint space, data acquisition during a movement; (5) basic gripper commands; (6) and a command returning the SCORBOT-ER 4u hardware to the home configuration.

```
% Initialize SCORBOT-ER 4u Hardware
ScorInit;
% Home hardware
ScorHome;

% Define desired waypoints as end-point XYZPR positions/orientations
XYZPR_i(1,:) = [500,-200,570,0,-2*pi/2];
XYZPR_i(2,:) = [500, 200,270,0, 0*pi/2];

% Convert waypoints to BSEPR
BSEPR_i(1,:) = ScorXYZPR2BSEPR( XYZPR_i(1,:) );
BSEPR_i(2,:) = ScorXYZPR2BSEPR( XYZPR_i(2,:) );

% Explore movement: Linear task movement with joint space waypoint
ScorSetBSEPR(BSEPR_i(1,:),'MoveType','LinearTask');
ScorWaitForMove;
% Explore movement: Linear joint movement with task space waypoint
ScorSetXYZPR(XYZPR_i(2,:),'MoveType','LinearJoint');
% Collect movement data
XYZPRs = [];
BSEPRs = [];
while ScorIsMoving
    XYZPRs(end+1,:) = ScorGetXYZPR;
    BSEPRs(end+1,:) = ScorGetBSEPR;
end
% Explore movement: Open the gripper using a character array
ScorSetGripper('Open');
ScorWaitForMove;
% Explore movement: Close the gripper using a scalar value
ScorSetGripper(0);
ScorWaitForMove;
% Return hardware to the home position
ScorGoHome;
ScorWaitForMove;
```

Beyond syntax matching, the ScorBot Toolbox simulation environment includes functions to further enhance the digital twin capabilities by providing context and expanded capabilities allowing the simulation to mimic the physical ScorBot-ER 4u setup at USNA. These functions include `ScorSimPatch`, `ScorSimLabBench`, `ScorSimDraw`, `ScorSimPlaceBlock`, `ScorSimGripBall`, `ScorSimCheckerBoard`, and `ScorSimGetSnapshot`. Figure 5, Figure 6, and Figure 7 provide examples of the context and capabilities added by these functions.

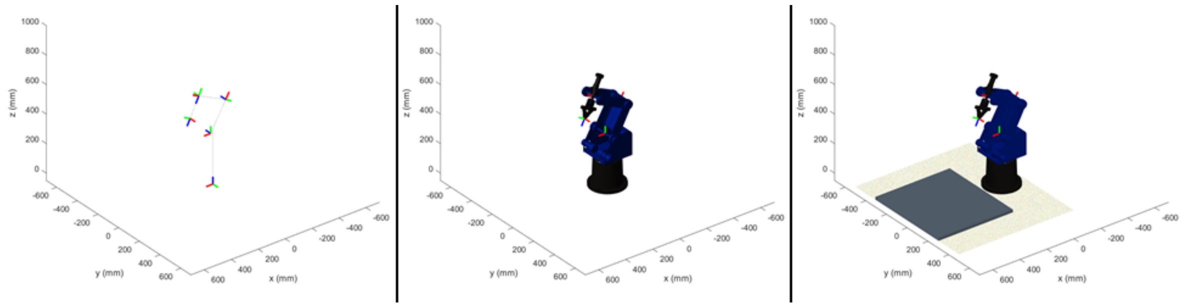

Figure 5: Basic kinematic simulation created using `ScorSimInit` (Left); rendered robotic components added to simulation using `ScorSimPatch` (Center); and lab bench and noisy background added to simulation using `ScorSimLabBench` (Right).

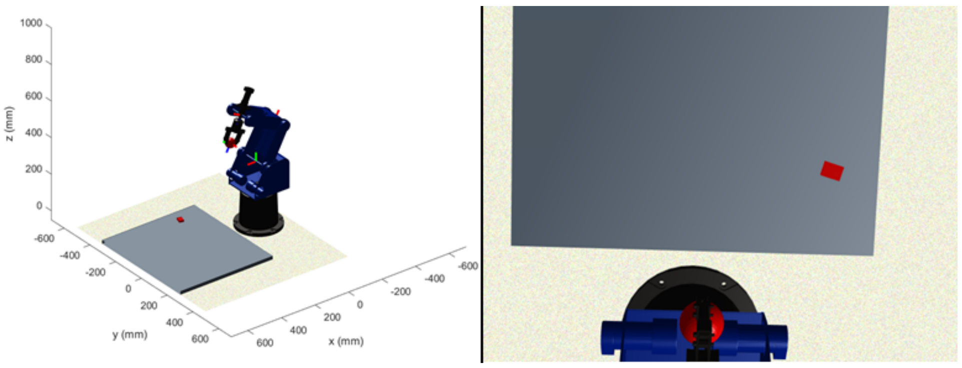

Figure 6: Rendered simulation and lab bench with block added using `ScorSimPlaceBlock` and gripped ball added using `ScorSimGripBall` (Left); and simulated image created using `ScorSimGetSnapshot` (Right).

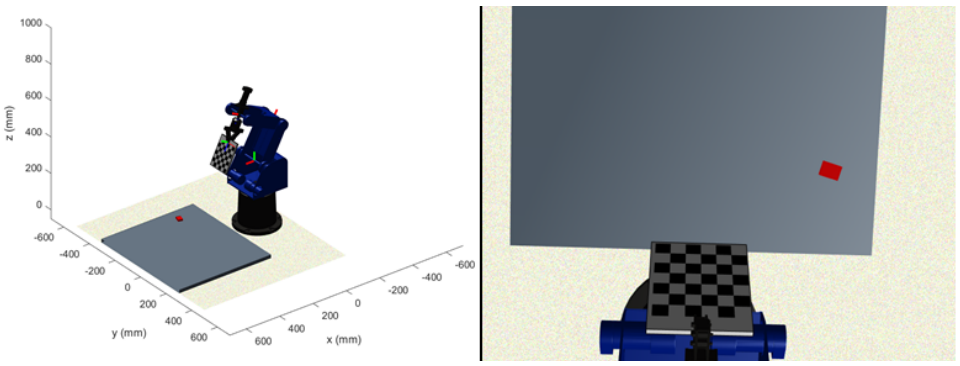

Figure 7: Rendered simulation and lab bench with block added using `ScorSimPlaceBlock` and gripped calibration checkerboard added using `ScorSimCheckerBoard` (Left); and simulated image created using `ScorSimGetSnapshot` (Right).

A specific example of these expanded digital twin capabilities is shown in the comparison between Figure 8 and Figure 9. Figure 8 shows the annotated instruction used as part of an in-person drawing exercise leveraging the SCORBOT-ER 4u hardware. Figure 9 shows screenshots of the equivalent, digital twin exercise used during remote learning. Note that Figure 9 uses the `ScorSimDraw` function included with the toolbox. In both hardware and simulation drawing exercises, students can: (1) create desired drawings relative to a paper-fixed coordinate frame, and (2) establish a transformation relating the paper-fixed frame to the robot base frame. These drawing tools are utilized in PBL exercises exploring inverse kinematics, linear movements in joint space, and the robot Jacobian.

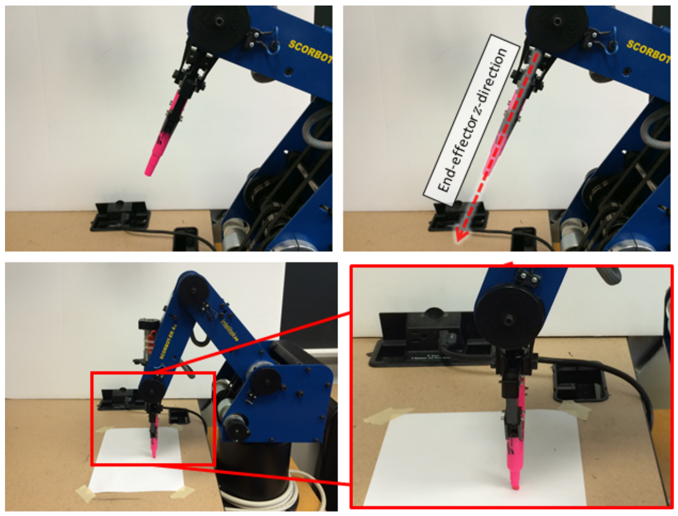

Figure 8: Annotated images of the SCORBOT-ER 4u used for instruction figures as part of a robotic drawing exercise using physical hardware.

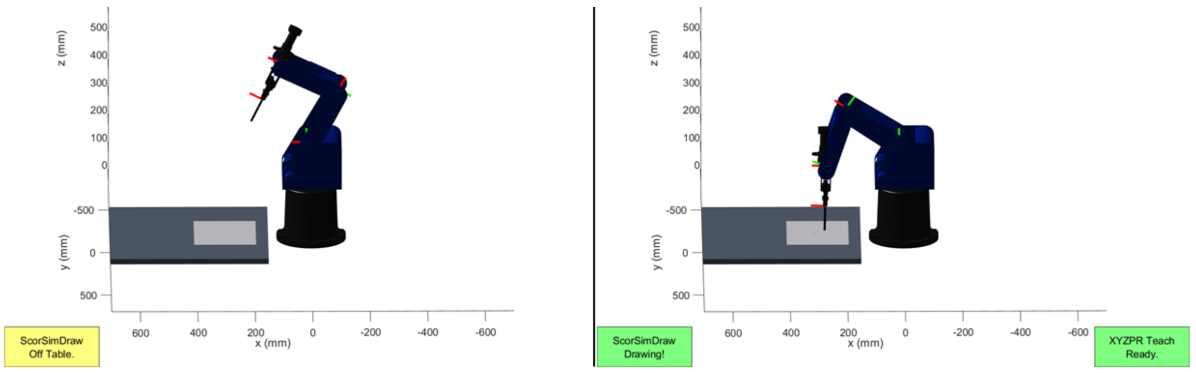

Figure 9: Screenshots of the ScorBot Toolbox simulation environment used for a simulated robotic drawing exercise during remote learning.

An additional example of the expanded digital twin capabilities is shown in Figure 10. In this example, students use simulated camera images to explore color-based image segmentation and binary object properties. Beyond fundamental image processing exercises, this simulated image capability can be used to explore robot/camera calibration, explore the use of camera feedback for grasp pose approximation, and explore applications of fixed-camera visual servoing.

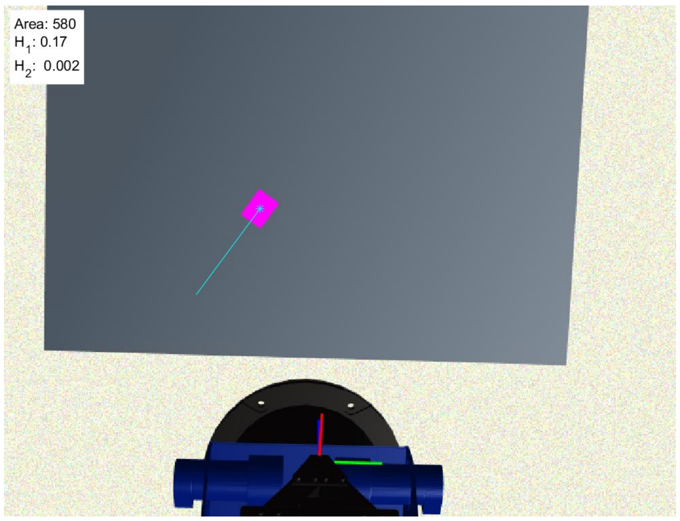

Figure 10: Simulated camera image of the ScorBot Toolbox simulation environment including rendered lab bench and block. This image includes an overlay highlighting the color-based segmentation of the block and the principal angle of the binary generated by the segmentation. Additionally, text is included in the upper left showing calculated values for the segmented object area, Hu's first moment invariant, and Hu's second moment invariant.

6 Discussion

The results presented in Section 5 provide a preliminary comparison between the capabilities of MTIS and the ScorBot Toolbox. This comparison is quantified using an analysis of the code included with each toolbox. Table 3 suggests that the ScorBot Toolbox is a more capable and complete tool for use with the SCORBOT-ER 4u. This is further reinforced by the efforts summarized in Section 3 and Section 4. Section 3 describes the efforts taken to create a consistent naming convention, while Table 1 and Table 2 concisely summarize the key functions for both hardware and simulation interactions. This directly addresses the imperfect results for the ease-of-use user survey presented in [2]. Section 4 describes the extensive work to handle errors, identify and eliminate bugs, and identify and eliminate crashes. This work directly addresses the imperfect results of the stability survey described in [2]. These efforts suggest that an equivalent survey conducted on the ScorBot Toolbox will result in improved student opinions when compared to MTIS.

The simulation tools and digital twin capabilities described in Section 5 provide a robust set of functions for use in a remote learning setting or as a complement to hardware interactions during in-person learning. These tools were used extensively as part of an Introduction to Robotics course offered to engineering students during the COVID-19 pandemic. During the of Fall 2020, courses at USNA began in a fully-remote setting and concluded with nearly three weeks of in-person learning. This unique transition meant that students interacted exclusively with simulation tools for the majority of the semester and were asked to complete a hands-on final project using the SCORBOT-ER 4u hardware. Even with no prior hardware experience, the similarity in syntax and extensive visualization tools included with the simulation environment gave students the necessary background to succeed in a hardware-based final project. Further, the anonymous, end-of-semester student opinions provided broadly positive feedback with statements including "[simulation] labs helped prepare for the final project," "[the] strongest features were the labs," and "the labs really were the strongest features of the course due to the practical application of what we were learning." Qualitatively, this suggests that the simulation tools included with the ScorBot Toolbox provide an effective digital twin for the SCORBOT-ER 4u hardware.

In general, the ScorBot Toolbox provides capabilities that complement existing coursework relating to robotics and computer vision. In an undergraduate setting, topics including rigid body kinematics and the pinhole camera model can provide powerful examples of the application of linear algebra while the robot Jacobian provides a straightforward application of partial derivatives. As a result, portions of the ScorBot Toolbox can be applied broadly across a variety of engineering disciplines to motivate core topics.

7 Conclusions and Future Work

This paper presents an open-source software tool developed for the Intelitek SCORBOT-ER 4u Educational Robot with a MATLAB front-end. The ScorBot Toolbox provides a documented, user-friendly, and open-source tool for installation, hardware interaction, kinematic modeling, visualization, and simulation using MATLAB.

The ScorBot Toolbox addresses the prescribed design goals through extensive work to ensure the usability, consistency, and stability of the software. As described in Section 2, the ScorBot Toolbox usability is enhanced using automated installation and version control, 64-bit support for the Windows OS enabling the use of the latest versions of MATLAB, and simulation tools that can be used on any MATLAB-compatible OS. Consistency is addressed using a simple naming convention summarized in Table 1 that allows users to easily identify applicable functions for given tasks. Lastly, Section 4 describes the extensive efforts to provide detailed error handling and descriptions, reduction of bugs, and reduction of crashes. The tools developed for this effort successfully eliminated the majority of issues present in MTIS and successfully eliminated errors associated with out-of-date and/or mismatched configuration libraries. Additionally, the `ScorSimTeachBSEPR` and `ScorSimTeachXYZPR` provide a teach pendant alternative to help reduce hardware crashes related to loose teach pendant cables.

In comparison to its predecessor, MTIS, the ScorBot Toolbox provides error handling, improved stability, simplified installation and updates, consistent function naming and syntax, extensive function documentation, and kinematic and 3D visualization simulation tools. Unlike MTIS, the ScorBot Toolbox supports hardware interaction on 32-bit and 64-bit Windows OS, and simulation tools are operational across all operating systems supported by MATLAB. Results presented in Section 5 suggest that the ScorBot Toolbox is a more capable and complete tool, and student opinions summarized in Section 6 suggest that the digital twin capabilities included with the toolbox provide an effective alternative for hands-on work with the SCORBOT-ER 4u.

Unlike [2], this effort has not included student surveys to quantitatively assess the ScorBot Toolbox in a classroom setting. While the anonymous, end-of-semester student opinions described in Section 6 provide broadly positive feedback that qualitatively suggests a positive assessment, future work will include an evaluation of the tool using structured surveys administered to a large sample of students taking robotics-related classes.

References

[1] "Scorbase and robocell: Robotic control and simulation software," accessed: Feb. 8, 2023. [Online]. Available: https://www.intelitek.com/resources/pdf/35-1007-4400_G_DS01_SW_Robocell-Scorbase.pdf

[2] J. Esposito, C. E. Wick, and K. A. Knowles, "A matlab toolbox for the usb intellitek scorbot," in *2011 ASEE Annual Conference & Exposition*, 2011, pp. 22–61.

[3] R. M. Felder and R. Brent, "Understanding student differences," *Journal of engineering education*, vol. 94, no. 1, pp. 57–72, 2005.

[4] J. C. Mosebo, "The scorbot-er 4u, function reference and notes for the usbc.dll," 2008, accessed: Feb. 8, 2023. [Online]. Available: https://www.theoldrobots.com/book45/USBC-document.pdf

[5] M. Kutzer, C. Wick, and J. Donnal, "ScorBot Toolbox for MATLAB," 3 2021. [Online]. Available: https://github.com/kutzer/ScorBotToolbox

[6] M. Kutzer, "Transformation Toolbox for MATLAB," 4 2022. [Online]. Available: https://github.com/kutzer/TransformationToolbox

[7] ——, "Geometry Toolbox for MATLAB," 3 2022. [Online]. Available: https://github.com/kutzer/GeometryToolbox

[8] ——, "Plotting Toolbox for MATLAB," 4 2022. [Online]. Available: https://github.com/kutzer/PlottingToolbox

[9] ——, "Patch Toolbox for MATLAB," 4 2021. [Online]. Available: https://github.com/kutzer/PatchToolbox

[10] ——, "WRC MATLAB Camera Support," 4 2022. [Online]. Available: https://github.com/kutzer/WRC_MATLABCameraSupport

[11] B. Mirko, "Error.h commenting included with scorbase 7.0.9.8," 6 1993, accessed: Feb. 8, 2023. [Online]. Available: https://downloads.intelitek.com/Software/Robotics/ER-4u/Previous_Versions/PLTW_Robocell_Scorbase_Setup_7.0.9.8.exe

[12] Shimon, "Error.h commenting included with scorbase 7.0.9.8," 9 2001, accessed: Feb. 8, 2023. [Online]. Available: https://downloads.intelitek.com/Software/Robotics/ER-4u/Previous_Versions/PLTW_Robocell_Scorbase_Setup_7.0.9.8.exe

[13] J. Esposito, "Download the toolbox, mtis files," accessed: Feb. 10, 2023. [Online]. Available: https://www.usna.edu/Users/weaprcon/esposito/_files/scorbot.matlab/MTIS.zip