applied sciences

Article

A System for Robotic Extraction of Fasteners

Austin Clark and Musa K. Jouaneh *

Department of Mechanical, Industrial and Systems Engineering, University of Rhode Island, Kingston, RI 02881, USA
* Correspondence: jouaneh@uri.edu

Abstract: Automating the extraction of mechanical fasteners from end-of-life (EOL) electronic waste is challenging due to unpredictable conditions and unknown fastener locations relative to robotic coordinates. This study develops a system for extracting cross-recessed screws using a Deep Convolutional Neural Network (DCNN) for screw detection, integrated with industrial robot simulation software. The simulation models the tooling, camera, environment, and robot kinematics, enabling real-time control and feedback between the robot and the simulation environment. The system, tested on a robotic platform with custom tooling, including force and torque sensors, aimed to optimize fastener removal. Key performance indicators included the speed and success rate of screw extraction, with success rates ranging from 78 to 89% on the first pass and 100% on the second. The system uses a state-based program design for fastener extraction, with real-time control via a web-socket interface. Despite its potential, the system faces limitations, such as longer cycle times, with single fastener extraction taking over 30 s. These challenges can be mitigated by refining the tooling, DCNN model, and control logic for improved efficiency.

Keywords: robotics; computer vision; DCNN; automation; disassembly

check for updates

Academic Editor: Pedro Couto

Received: 4 October 2024
Revised: 22 December 2024
Accepted: 8 January 2025
Published: 10 January 2025

Citation: Clark, A.; Jouaneh, M.K. A System for Robotic Extraction of Fasteners. *Appl. Sci.* **2025**, *15*, 618. https://doi.org/10.3390/app15020618

1. Introduction

There is a growing need for more intelligent and adaptive industrial robot capabilities within robotic work cells in scenarios regarding servicing, managing EOL components, and reclaiming e-waste materials in a non-destructive manner. A vast amount of preliminary research into this field focuses on feasibility assessments of product disassembly for component re-purposing or recovery by analyzing product waste streams and the complexity of high-value component assemblies [1,2]. Other areas of focus are various non-destructive or semi-destructive initiatives for reclaiming materials from EOL electric vehicle waste streams [1,3,4]. Some methods proposed include variations in human–machine collaboration for EOL battery recycling where humans can handle non-trivial tasks while collaborating robot platforms can manage high-volume repetitive taskings like removing screws and bolts [4,5]. Other areas of automated disassembly focus on consumer products like LCD screens [6,7], which, given the lack of predetermined knowledge of fastener position, still rely on semi-destructive methods like using angle grinders and power drills to separate components.

Mechanical fastener extraction constitutes approximately 40% of all disassembly operations [8]. Several researchers have addressed the automated disassembly of fasteners. DiFilippo and Jouaneh [9] developed a system to remove screws from the backs of laptops automatically using non-AI vision algorithms. Li et al. [8] presented a device for automated hexagonal-headed threaded fastener removal that used a spiral search strategy to overcome uncertainties in the location of the fasteners. Chen et al. [10] introduced a method that utilized images of a screw to assist in aligning the screw–tool interaction, with torque monitoring employed throughout the screw removal process.

With recent advances in vision-based AI in the classification and accurate identification of mechanical components using varying region-based convolutional neural networks (R-CNNs) and DCNN methods, many researchers [11–19] have reported on the use of vision-based AI in screw detection. This makes it possible to integrate this technology with robotic systems for applications such as reclaiming e-waste components. This task is traditionally prohibitively expensive with human operators and needs to be more efficient when relying on existing automated methods.

While there is a significant interest in the use of AI methods for screw detection, only a few studies [20,21] have addressed the development of AI-based robotic screw disassembly systems. This paper focuses on developing a robotic system capable of extracting cross-recessed mechanical fasteners through an application programming interface (API) layer. This system integrates a standalone DCNN vision system, explicitly trained to detect cross-recessed screws, with robotic simulation software designed for industrial robot applications. The system was tested on cross-recessed screw (CRS) targets with a 0.5-inch (12.7 mm) diameter at camera ranges of 280–500 mm. It employs a stereoscopic camera system with an RGB (red, green, blue) imager. The primary contribution of this work lies in the integration of a DCNN vision-based system, simulation software, and robotic framework for screw removal, as well as the development of a two-stage imaging process for screw disassembly. Unlike previous work [20,21], which focused on the disassembly of hexagonal-headed screws, this research explicitly targets CRS disassembly. Figure 1 illustrates the input images used to train this study's DCNN implemented for fastener detection.

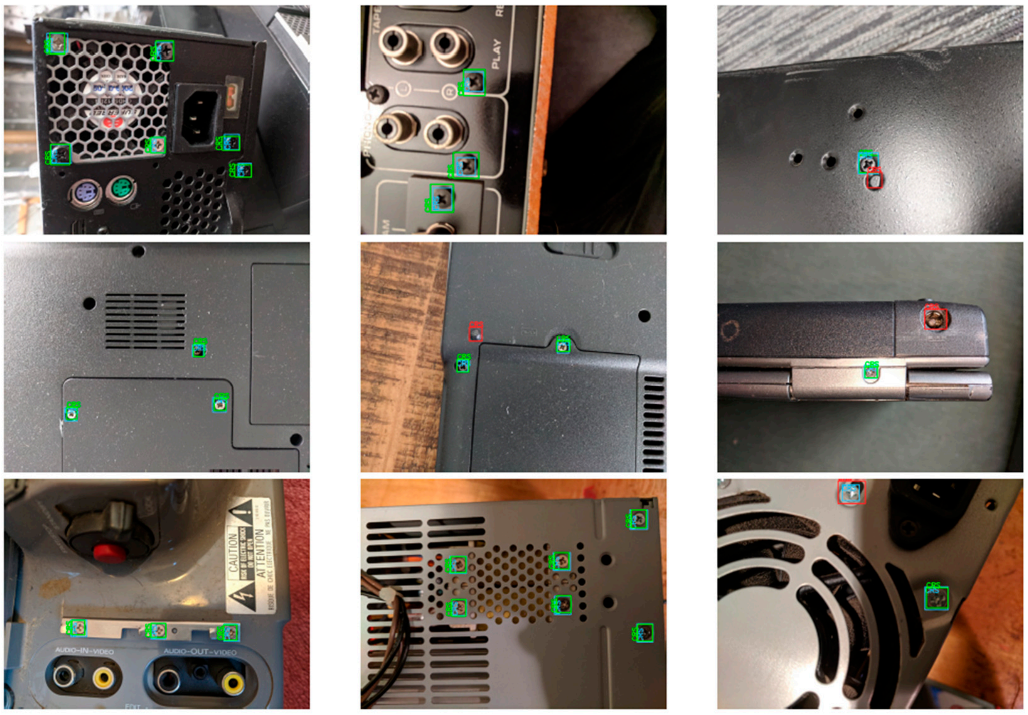

Figure 1. Training images for the classification of cross-recessed screws [12].

The remainder of this paper is organized as follows. Section 2 of this work focuses on methodology, including introducing the simulation software, and how other software was integrated to provide detection and fastener removal capabilities. Section 3 provides a system overview, which incorporates a discussion of the design of the hardware and software components implemented and introduces some of the mathematical and robotics concepts for the problem domain. Within Section 3, an introduction of two of the state-machine-based extraction methods tested throughout the evaluation phase in this work is provided. This is followed by a brief discussion of managing object detections and how that pertained to the output of the DCNN predictions to the video stream provided by the robot's tooling. The evaluation in Section 4 provides a single sample extraction plot that incorporates readings from the custom tooling integrated with current and force sensors, a state sequence index plot, and readings from the robot position and camera detections. Bulk run data against the testing artifact introduced later in this work are provided to give a general sense of the system's capability with respect to two of the extraction processes designed and used in this paper. A conclusion is provided in Section 5 regarding the evaluation and discussion of the aggregate data from bulk CRS extraction testing and what was accomplished in this work.

2. Methodology

In this work, we used the RoboDK software (v5.5.2) as the simulator for our system. RoboDK is software for simulating, programming, and implementing program logic for industrial robots before deploying to physical hardware. This software platform allows users to create and simulate robotic operations for various robot brands and applications, and it also generates robot programs for the specified robots, thereby reducing downtime and improving productivity. With its extensive library of robot models and post-processors, RoboDK is easily adaptable via a standard API to different robot controllers and platforms, with minimal changes to the source code necessary to retrofit.

RoboDK also supports online programming through its user interface, which allows users to control their robots in real time, adapt to changing manufacturing requirements, and troubleshoot any issues that may arise during functional testing. Leveraging the online programming functionality and extensive API, users can stream the same code used for offline testing in a physical simulation directly to the robot's controller via one of the software libraries of drivers. Figure 2 illustrates the two communication methods for interfacing with physical robot hardware. The online programming method in the right half of the figure is primarily employed throughout this work.

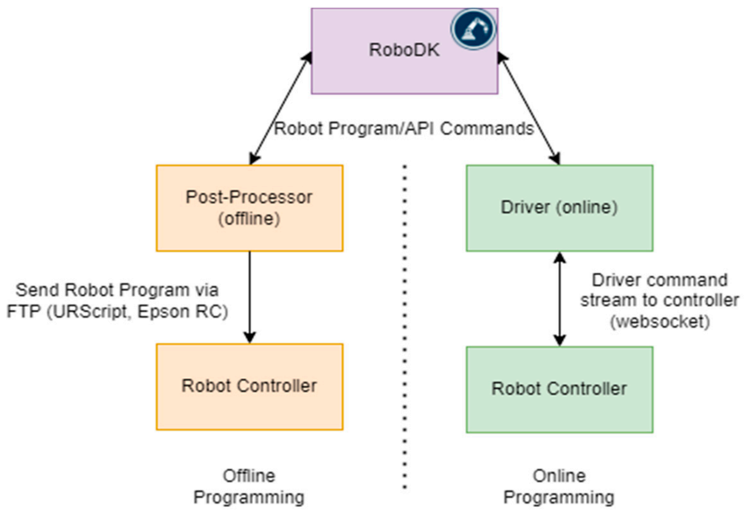
RoboDK
Robot Program/API Commands
Post-Processor (offline)
Driver (online)
Send Robot Program via FTP (URScript, Epson RC)
Driver command stream to controller (websocket)
Robot Controller
Robot Controller
Offline Programming
Online Programming

Figure 2. RoboDK communication channels for executing robot programs on physical hardware.

In this project, the simulated robot continuously received commands, such as inverse kinematic endpoints, movement constraints (e.g., linear or joint move limits), tool change commands, and speed commands, through the API. These commands were converted in real time into driver calls, synchronizing the movements of the physical system. The driver also provided feedback such as error states, driver status, and real-time data like the actual robot position. This feedback integrated into the state logic, enabling the system to queue commands (to prevent sending a new move command while others are pending) and facilitate decision-making in the state-based program logic. For instance, the system can avoid specific joint movements based on constraints set on the simulated robot's joints to prevent potential collisions. The primary functionalities that enable this simulation suite's capabilities are given below.

- Ease of use through extensive API (C#, Python (3.7.8), MATLAB (R2022a));
- Robotics tools (kinematic model, trajectory planning, pose transformations);
- Comprehensive post-processor system (offline programming);
- Extensive driver support for major brands of industrial and collaborative robots (online programming).

The physical system to be simulated consisted of a UR5 robot equipped with task-oriented tooling and stereoscopic camera equipment. Figure 3 shows the physical system and its simulated version in RoboDK.

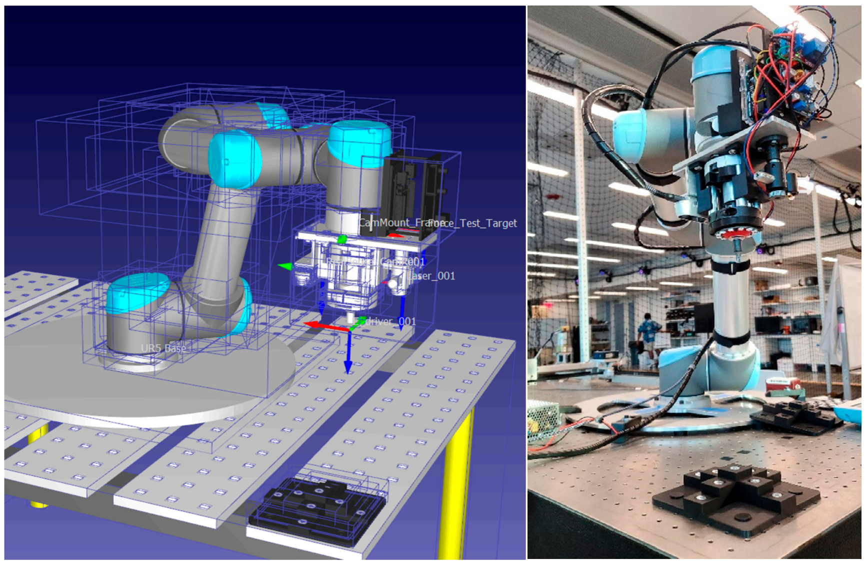

Figure 3. (**a**) RoboDK fastener extraction simulation. (**b**) UR5 fastener extraction platform.

The simulated variant of the UR5 is shown with collision detection boundaries, which, when executing moves, would predetermine and halt the online robot if a collision was estimated. Joint limits were implemented at the shoulder and wrist joints within the simulation to provide only kinematic solutions that would not be obstructed by the robot's wrist-mounted electronics assembly or cable spine. In addition, the RoboDK simulation suite employs the UR5's Denavit–Hartenberg parameters for an accurate kinematic model. These parameters are used for analytical kinematic solutions to the tool center points (TCPs) defined on the tool carriage in the simulation within Figure 4.

A tool assembly was designed to carry an electric screwdriver, a stereoscopic camera, and a cross-profile laser and mount assembly. A compact suite of electronics (not shown) was also implemented for ingesting incoming control signals, providing signals to the motor and adjacent tools, and measuring compressive force and torque experienced by the tool during operation to be returned to the control computer. A view of the completed tool assembly is shown in Figure 5a, followed by an exploded assembly view in Figure 5b. Figure 4 shows the tool frame definitions of the tool assembly in the RoboDK software. Three frames were defined in the RoboDK simulation application with respect to the CAD geometry. These were the laser, the electric screwdriver, and the camera frame. The frame definition of the camera was placed with respect to the RGB imager origin (as defined by the Intel RealSense D435 camera datasheet) and matched the camera coordinate system. The pose of the camera imager frame was stored by the camera streaming script for deprojection with each instance of recorded target positions in the camera frame.

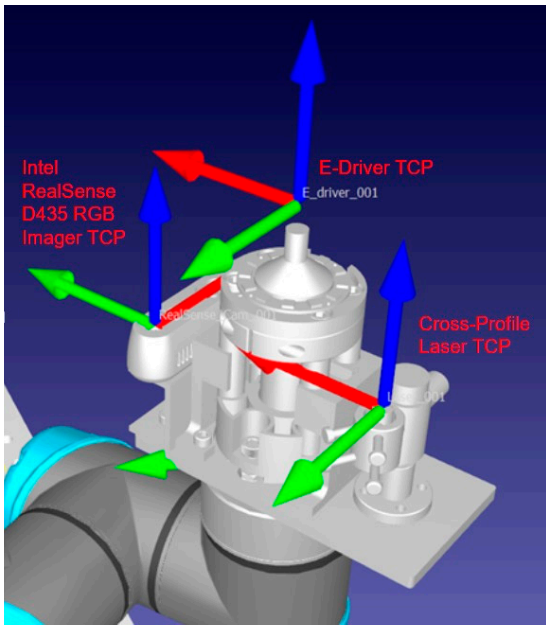

Figure 4. Simulation suite CAD-based TCP definitions of tools.

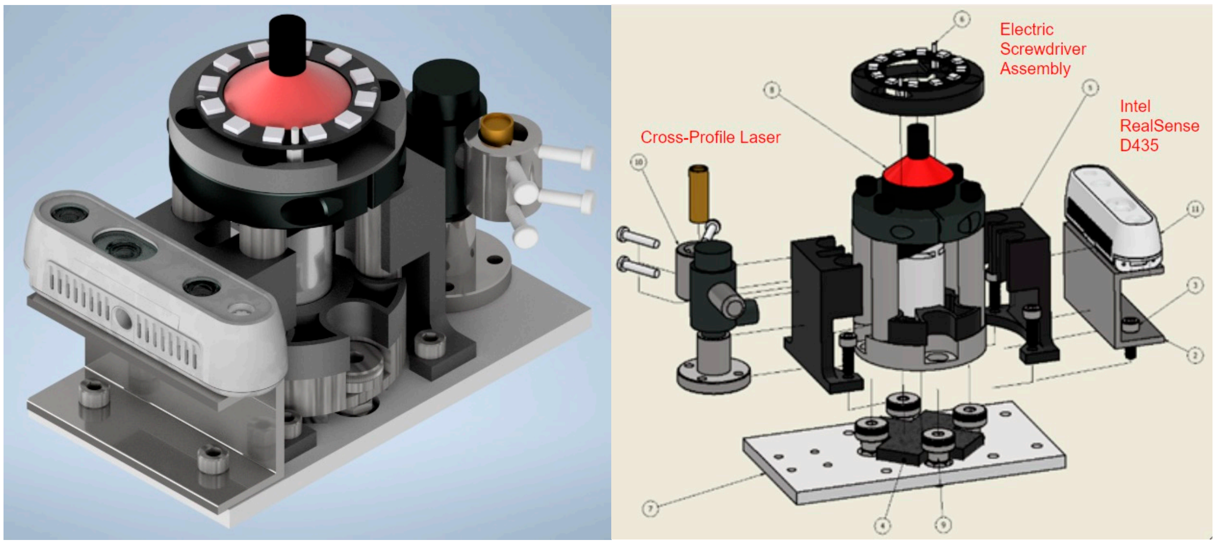

Figure 5. (**a**) Isometric render of tool assembly. (**b**) Exploded view of tool assembly.

RoboDK was chosen as the foundation for the simulation software in this work. The primary reasons for its selection include its support for online driver programming, built-in driver support for the UR product line, extensive documentation of its Python API, and a cost-effective licensing scheme suitable for academic applications. Similarly, the Intel RealSense D435 was selected over various industry stereoscopic camera systems based on several criteria: an easily implementable and open-source programming API, a calibration suite, widespread academic use, extensive documentation, and a compact form factor. The DCNN was employed within the tiny-yolo v2 framework using a model we previously developed [12]. It was chosen for its compatibility with the existing Python version that constituted most of the project's code base rather than being retrained on a newer model. The DCNN model we used yielded an average precision (AP) between 92.6% and 99.2% for the different testing datasets. Further information on the DCNN model and the training details are discussed in [12]. The UR5 robotic arm was utilized as an available laboratory asset. Still, the project's code base was designed to leverage RoboDK's features, enabling the substitution of the UR5 in the software suite with a different 6-axis robotic arm, incorporating new geometry, drivers, and a kinematic model in the future.

Additionally, tool assembly components, such as the laser assembly, electric screwdriver, and controller electronics, were either custom-made or employed educational-grade components (as opposed to industry-grade) for cost efficiency. Hardware solutions for power conversion, force and torque sensing, switching, and control were custom-implemented using consumer off-the-shelf components such as buck converters, relays, a microcontroller, and a motor driver shield.

3. System Overview

3.1. Camera Relationships

As an integration project, this work focuses on digitizing and implementing existing concepts to design a process and augment the automated CRS extraction system to achieve desirable and consistent behavior. The discussion in this section uses mathematical concepts described in various robotics modeling texts, such as the Siciliano text [22]. Many of the mathematical concepts presented in this section were applied using multiple software APIs such as the Intel RealSense pyrealsense2 package and RoboDK's robomath package.

In this work, O_B represents the base coordinate frame for the robot, O_C represents the coordinate frame for the camera system, and O_O represents the target coordinate frame. In addition to these coordinate frames, a world frame, O_W, exists in which all other coordinate frames can be defined in absolute coordinates with respect to one another. Note that the world frame exists at an arbitrary and stationary point in the environment differing from the robot base frame. Figure 6 shows the pinhole camera model. This model describes the mathematical relationship between the coordinates of a point in three-dimensional space and its projection onto an image plane.

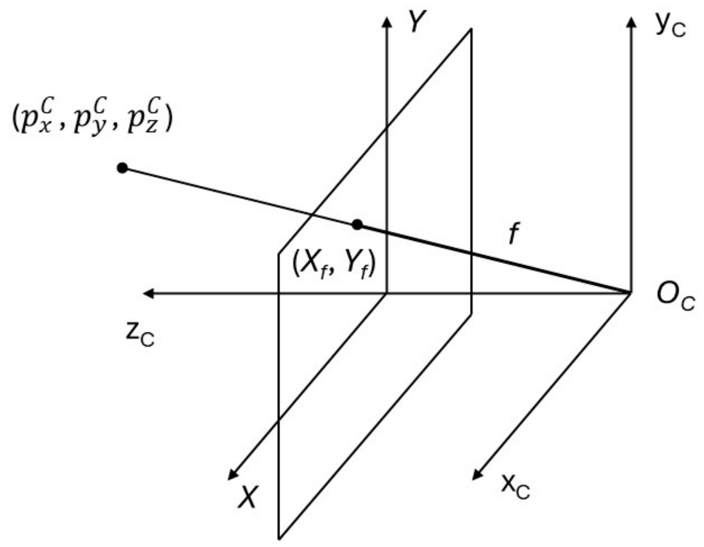

Figure 6. Frontal perspective pinhole camera model for coordinate deprojection.

This work focuses on inverting the generic pinhole camera model relationship and deprojecting points from an image projection. Deprojection is the act of taking points from a camera's 2D image perspective and computing their location in 3D space using camera properties, the image coordinates of said points (in pixels), and relative distance measurements.

$$p_x^C = \frac{(X_I - X_O)p_z^C}{\alpha_x f} \tag{1}$$

$$p_y^C = \frac{(Y_I - Y_O)p_z^C}{\alpha_y f} \tag{2}$$

Equations (1) and (2) give the x and y coordinates, respectively, of an object in the camera frame, given the focal length of the camera (f). α_x and α_y represent the pixel-to-spatial sampling unit conversion (between pixels and millimeters) factors based on the camera's intrinsic properties. The size of the image (in pixels) and its relationship to the center point of the image plane are given by the expressions $X_I - X_O$ and $Y_I - Y_O$ for the x and y coordinates, respectively. Next, we form the 4×1 augmented target vector given by Equation (3), in which p_x^C and p_y^C are obtained from deprojection methods, and p_z^C is obtained from the camera stereoscopic sensors which define the object distance with respect to the camera frame O_C. To obtain the coordinates of the object in the world reference frame, Equation (5) is used to multiply the P_O^C vector by the 4×4 homogeneous transformation matrix given by Equation (4), which describes the position and orientation of the camera frame in the world frame.

$$P_O^C = \begin{bmatrix} p_x^C \\ p_y^C \\ p_z^C \\ 1 \end{bmatrix} \tag{3}$$

$$T_C^W = \begin{bmatrix} R_C^W & P_C^W \\ 0\,0\,0 & 1 \end{bmatrix} \tag{4}$$

$$P_O^W = T_C^W P_O^C \tag{5}$$

$$P_O^B = {T_B^W}^{-1} T_C^W P_O^C \tag{6}$$

Alternatively, Equation (6) can be used to represent the target P_O^C with respect to the base of the robot P_O^B, where T_B^W is the transformation matrix between the base frame and the world frame. This representation of the coordinates is ideal in software and can be used to issue open-loop move commands to an inverse kinematics endpoint.

3.2. Software Design

The deployment of this system required the development of a Python package capable of interfacing with all incorporated hardware and software components. This Python Package element consisted of numerous Python modules for any operational instance of the system, as shown in Figure 7. For details, see [23]. The computer vision components (OpenCV and the Intel RealSense python API) of this package interface with the external CRS Detection package via the usage of the TFNet methods (which expose the frame predictions of the tiny-yolo-v2 DCNN vision system). In addition, several robot development kit (RDK) station files were developed to execute various capabilities and test cases within this software package. Several batch scripts were created to streamline the process of launching everything, as multiple Python processes are necessary at any given time when the system is running. Lastly, a state-machine-based microcontroller program is stored within the automated CRS extraction software package, which must be compiled and uploaded to the microcontroller unit (MCU) before the system can operate. This system ultimately relies upon the various RoboDK Python APIs to interface with the simulation environment and communicate via the simulation software suite to the physical robot controller via a robot driver.

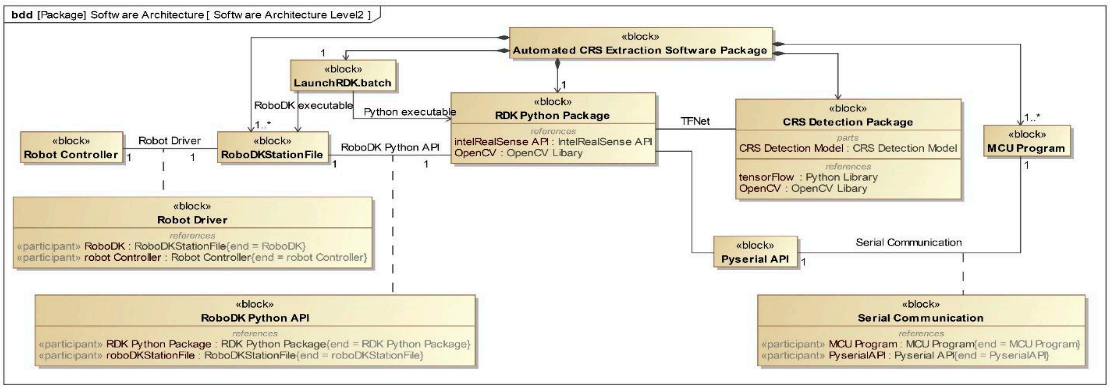
bdd [Package] Software Architecture [Software Architecture Level2]
«block» Automated CRS Extraction Software Package
«block» LaunchRDK.batch
«block» RDK Python Package
references
intelRealSense API : IntelRealSense API
OpenCV : OpenCV Libary
«block» CRS Detection Package
parts
CRS Detection Model : CRS Detection Model
references
tensorFlow : Python Library
OpenCV : OpenCV Libary
«block» MCU Program
«block» Robot Controller
«block» RoboDKStationFile
Robot Driver
RoboDK executable
Python executable
RoboDK Python API
TFNet
«block» Robot Driver
references
«participant» RoboDK : RoboDKStationFile{end = RoboDK}
«participant» robot Controller : Robot Controller{end = robot Controller}
«block» Pyserial API
Serial Communication
«block» RoboDK Python API
references
«participant» RDK Python Package : RDK Python Package{end = RDK Python Package}
«participant» roboDKStationFile : RoboDKStationFile{end = roboDKStationFile}
«block» Serial Communication
references
«participant» MCU Program : MCU Program{end = MCU Program}
«participant» PyserialAPI : Pyserial API{end = PyserialAPI}
1
1..*

Figure 7. High-level software dependencies and architecture used to interface with robot hardware.

Once fully integrated with all hardware, several extraction methods were defined and are outlined in this section using state-based behavioral diagrams. The methods vary in how the images of the targets are acquired and handled throughout an extraction attempt to provide the most accurate open-loop move commands for the tooling. Figure 8 depicts the primary state machine governing logic, which uses a two-stage imaging process for target deprojection and globalization. Table 1 provides the complete non-abbreviated state transition behavior. This process provided higher successful extraction rates after rigorous stereoscopic camera deprojection testing and an initial single-stage imaging and extraction process. The comparison between the more basic and two-stage methods is outlined below, and a single sample extraction run is presented later in this section.

Table 1. State transitions corresponding to the callouts in Figure 8.

Condition	Meaning
a	Target(s) acquired
b	All targets extracted (or attempted)
c	The current pose does not equal the staging pose
e	Laser position reached/robot driver not busy
f	Electric screwdriver position reached/robot driver not busy
g	Laser pose inaccurate (user interaction)
h	Electric screwdriver position reached/robot driver not busy
i	Incremental robot move completed/robot driver not busy
j	Force-sensing resistor (FSR) conductance threshold reached or exceeded
k	Current threshold (in the motor driver board) reached
l	Tool commands sent/command timeout reached
m	Tool ramp-up timer completed
n	Current threshold has not been reached
p	FSR conductance threshold not reached
q	All incremental moves completed
r	Electric screwdriver pose inaccurate (user interaction)
s	New target coordinates acquired from NN
u	Wait interval elapsed

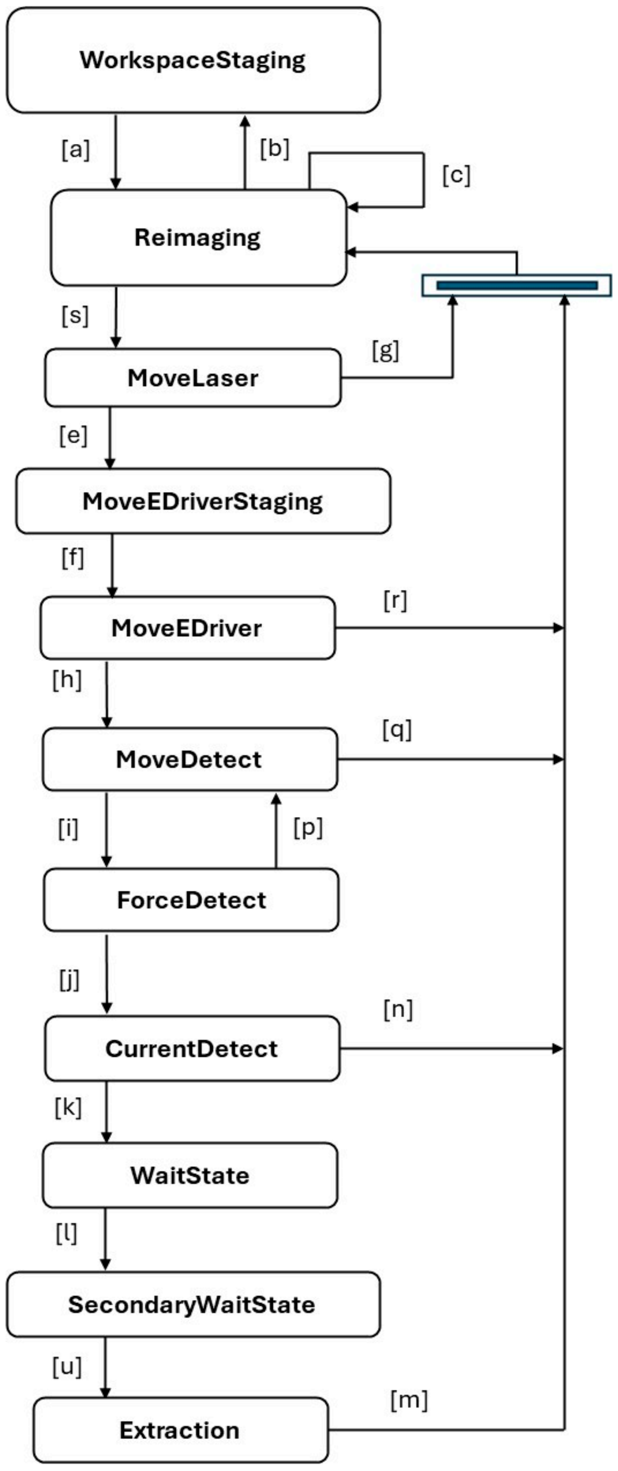

Figure 8. Method 2 state machine for guiding the UR5 to extract CRS targets.

The two methods used were as follows:

Method #1—Single-Shot High-Level Imaging with DCNN

- A single high-level image is used for all target acquisitions. The image is taken with a stereoscopic RGB imager positioned further than 350 mm and roughly positioned above the workpiece such that all cross-recessed screws are in the field of view (FOV).

Method #2—Two-Level Imaging with DCNN

- A first level of DCNN imaging is used initially for the approximate locations of targets (from height > 350 mm). These coordinates center the camera package's RGB imager directly over the target.
- A second level of DCNN imaging is used with the RGB camera centered over the target. The second imaging stage is used at a height of less than 320 mm and greater than 300 mm (as close as possible with stereoscopic sensors still functioning accurately).

A challenge encountered in this work was the need to manage object detections accurately within the robot's workspace. The DCNN provided a continuous stream of object detections at a consistent update interval (every 3–5 s, depending on the operational mode). It was essential to implement a logic that prevented the addition of redundant target datasets for negligible changes in detected x, y, or z coordinates, as reported by the camera sensors or model predictions. This issue became particularly pronounced when the robot moved, requiring the recalculation of globalized coordinates based on the new camera position. Figure 9 illustrates an example of DCNN detections and stereoscopic camera coordinates.

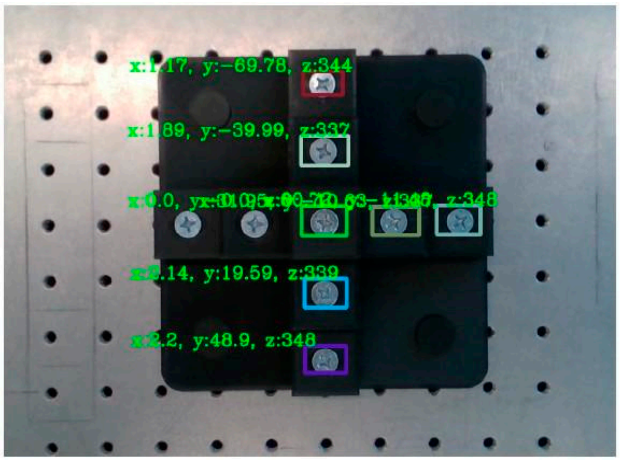

Figure 9. Sample of DCNN detections and stereoscopic camera coordinates.

To address this, a Python module was developed to interact with the existing system used for reading and globalizing target coordinates from the camera streaming module. This module interfaced with the RoboDK Python API to relay commands and defined a tolerance region for each discrete detection. The tolerance region prevented redundant entries by overwriting detections that were approximately located within the same region. Only one set of unique coordinates was stored at any given time. The module also supported functionality to compute a rolling average of previous coordinate sets from the intermediate file in which the Python streaming module stored detection data.

The tolerance region was implemented as a rectangular band, typically defined as 5–15 mm in the x and y dimensions, centered at the initial detection point of a unique object. Any subsequent detections within this band were considered redundant and not added as new targets. Detections falling outside the tolerance band were classified as new targets and appended to an array of unique detections for further processing.

The approach proved effective for managing object detections in this application. However, in other applications with a highly dynamic environment where targets frequently change position or overlap, our approach may falter due to its reliance on fixed tolerance bands and sequential processing. A possible solution for this issue would be implementing adaptive tolerance bands that adjust based on environmental conditions or object distribution.

4. Results and Discussion

Figure 10 shows a sample extraction run using method #2's extraction process, which correlates to a single execution of the state machine in Figure 8. This Figure gives time series data for the force sensor, current sensor, active state, and TCP z-coordinate tools.

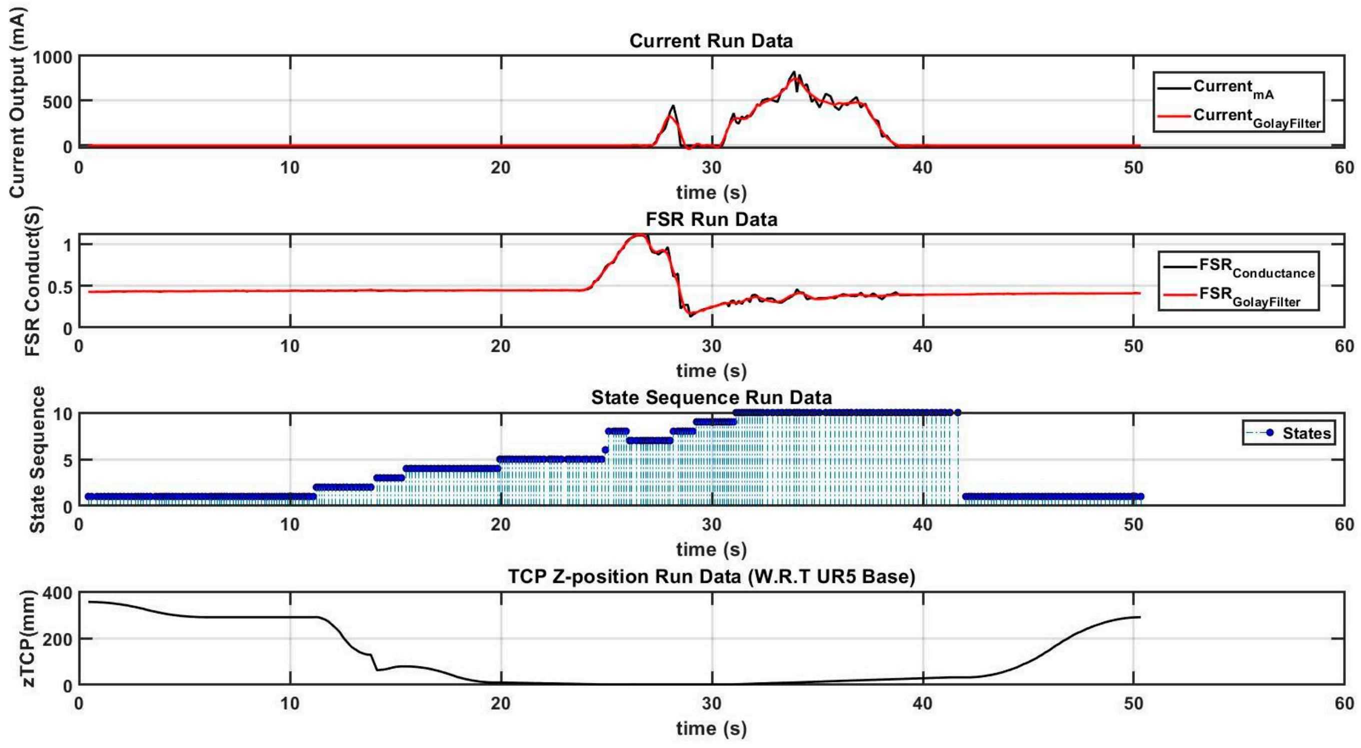

Figure 10. Extraction run data executed under the logic outlined in Figure 8.

In Figure 10, the top subplot illustrates the current data in milliamps (mA) as recorded by the motor driver board during the fastener extraction process. Two distinct peaks are visible. The first peak represents the initial current detection state, where a low RPM spin is used as the tool lowers to the known position of the CRS to check for thread engagement. This peak, reaching nearly 500 mA, indicates the tool's recognition of engagement with the fastener profile, triggering the screwdriver to shut down promptly. The second peak illustrates the motor current fluctuations as the tool loosens and disengages the fastener during the extraction phase. The second subplot displays the Force-Sensitive Resistor (FSR) data, which measures force as the tool compresses and contacts the testing artifact. An initial high peak at the 25 s mark indicates the first contact, followed by a slight decrease in force and a plateau around the 27 s mark. This change occurs as the tool bit engages with the screw threads, and slightly jogs upwards to reduce the force on the fastener. The sharp decline and fluctuating readings between 30 and 40 s mark the extraction phase, where the pressure is relieved as the tool unthreads the fastener and gradually ascends in the *z*-axis. The third subplot shows the state index time series data, where individual integers represent discrete states, as detailed in Table 2. The final subplot displays the z-position of the active tool, as reported by the robot driver and the known TCP in the RoboDK software. Here, the *z*-axis is oriented relative to the robot base and decreases to the second imaging state z-coordinate, as defined in Table 3. The abrupt changes in the *z*-axis observed between the ten- and twenty-second marks correspond to the software switching the active tool between the laser, RGB camera, and electric screwdriver.

Table 2. State sequence index plot key.

State	Sequence Key
1	Reimaging
2	MoveLaser
3	MoveEDriverStaging
4	MoveEDriver
5	MoveDetect
6	ForceDetect
7	CurrentDetect
8	WaitState
9	SecondaryWaitState
10	Extraction

Table 3. Sample target and run parameters associated with the run data in Figure 10.

Target Parameters	x-Coord (mm)	y-Coord (mm)	z-Coord (mm)	Confidence (%)
Initial Imaging State	571.32	−401.71	3.09	0.75
Second Imaging State	570.15	−402.60	2.05	0.77
Run Parameters	Detection Conductance Threshold (S)	Detection Current Threshold (mA)	Maximum Allowable Current (mA)	Maximum Detected Current (mA)
Run Values	0.5153	350	2000	909

The cycle time for the run shown in Figure 10 is 50.5 s, which is consistent for each run (almost all runs took less than 55 s for each target). Roughly 40% of the time in this method is used for re-imaging the desired target and traversing to and from the robot's staging position. This cycle time can be reduced in future iterations of this system by (a) using higher robot speed during the traverse back and forth and screw extraction phases, as the speed was conservatively kept low due to early-stage testing, (b) replacing the force-sensing resistor with a force/torque sensor which gives more reliable force detection and a faster response, and (c) deactivating the laser on the tool assembly which only provides a visual queue of the intended target and its accuracy but is not used in the logic for the extraction process.

As previously defined, Table 2 represents the states on the *y*-axis of the State Sequence Run Data subplot in Figure 10. While the logic for the state program controlling the robots' actions is continuously re-executing, the time series data of which state the robot is currently in are continuously aggregated for status monitoring. Table 3 represents additional operational parameters collected on each CRS extraction attempt. These data include the relative current and conductance thresholds while the physical tooling executes, the initial and re-acquired target x, y, and z globalized coordinates, and the DCNN detection confidence threshold.

Figure 11 shows the nine targets on the testing artifact used in this work. Experiments were conducted with one type of fastener material to minimize possible surface reflection effects on detection. Table 4 gives a sample single run for single-shot high-level imaging of all targets within the robot's workspace. Three failures were measured from this sample extraction process, which, unlike the process outlined in Figure 8, omitted any form of camera repositioning for more optimized target deprojection. Targets 1, 2, and 9 were noted as failing due to general inaccuracy, causing threads to not engage during the CurrentDetect state in which the tool attempts to tighten the CRS to detect thread engagement and properly seat the TCP if necessary. The overall first-pass yield for this sample run was 66.67%.

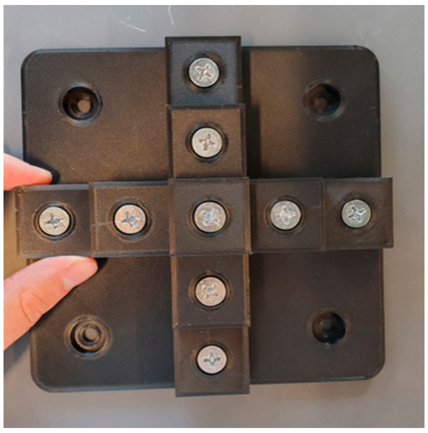

Figure 11. The nine (9)-target symmetrical stepped testing artifact for CRS extraction.

Table 4. Bulk run data using single-stage imaging with DCNN.

Target Number	Successful Extract FP	Image Number	Coordinates (mm)			Confidence (0.00–1.00)
			x	y	z	
1		First	566.79	−402.16	4.94	0.85
2		First	537.34	−401.78	13.92	0.89
3	X	First	507.56	−431.97	13.94	0.70
4	X	First	477.83	−402.49	10.94	0.86
5	X	First	447.79	−403.41	0.00	0.84
6	X	First	508.70	−463.84	−0.04	0.80
7	X	First	507.64	−402.19	23.94	0.84
8	X	First	507.22	−344.28	2.91	0.47
9		First	507.21	−372.43	11.92	0.70

Table 5 presents a sample of a bulk testing run within the robot's workspace. The sample extraction data are given for the nine targets on two extraction passes. After attempting each target, if the vision system detects the remaining targets, it will execute another extraction attempt on them. It should also be noted that targets 1–9 are not in any order (left to right or top to bottom) for the sample extraction run. This is because each coordinate was set initially at random until all targets were detected, and the vision system updated the targets according to a predefined global tolerance.

The two sets of coordinates for each target represent the globalized coordinates from the initial workspace staging state and the re-imaging state for the first and second image numbers, respectively. These coordinates are for the extraction that succeeded, so if it failed on the first pass, both sets of coordinates are for the second pass. The "X" in the Successful Extract columns indicates that the cross-recessed screw extraction was successful on that pass. Success is defined as the CRS threads fully disengaging from the heat-set threaded inserts on the testing artifact.

The FP and SP columns indicate "first pass" and "second pass" exclusively, referring to Table 5. The total FPY (first-pass yield) ranges from 78% to 89%, and the total SPY (second-pass yield) is 100% in all three instances. An interesting trend that can be observed from all method #2 bulk run data and in Table 5 is that of the confidence interval generally increasing or staying the same with the second pass, which coincides with the camera and DCNN system overall performing better (more consistent deprojection behavior) when the RGB imager is directly in line with the target. This is evident with targets 1, 2, 4, 6, and 8.

Table 5. Bulk run data using two-stage imaging with DCNN.

Target Number	Successful Extract FP	Successful Extract SP	Image Number	Coordinates (mm)			Confidence (0.00–1.00)
				x	y	z	
1	X		First	418.38	−402.66	2.63	0.28
			Second	417.48	−402.96	2.07	0.53
2	X		First	387.57	−403.15	10.12	0.85
			Second	388.12	−402.58	10.62	0.88
3	X		First	359.59	−462.72	−0.39	0.82
			Second	357.83	−463.26	1.03	0.79
4	X		First	356.65	−372.46	11.23	0.55
			Second	357.01	−373.06	12.78	0.63
5	X		First	357.51	−401.24	22.72	0.80
			Second	358.85	−402.51	20.53	0.36
6	X		First	358.90	−432.23	11.51	0.56
			Second	358.89	−432.25	11.51	0.56
7	X		First	328.15	−402.30	11.62	0.61
			Second	328.12	−401.88	10.69	0.88
8	X		First	297.11	−403.26	0.14	0.47
			Second	298.83	−402.65	3.63	0.76
9		X	First	358.31	−340.42	3.49	0.32
			Second	358.27	−340.37	3.49	0.32

Table 6 compares the results obtained from this work with robotic screw disassembly systems reported in the literature. Note that references [20,21] address the removal of hexagonal-headed screws, while our work targets CRSs. All the works in the table have some form of force or torque sensing to help the tool engage with the screw. In reference [20], a 100% extraction accuracy rate is attributed to the combined use of a deep learning model, a force/torque-based analytical model for the tool–screw interaction, and an optimization algorithm.

Table 6. Comparison of robotic screw disassembly systems.

Reference	Type of Fastener	Sensing Methods	Extraction Accuracy
Peng et al. [20]	Hexagonal bolts	Camera and Force/Torque sensor	100%
Zhang et al. [21]	Hexagonal bolts	Camera and built-in torque sensing in the UR robot	Not reported, but they reported a 100% success rate in the vision-based pose estimation
This work	Cross-recessed screws	Camera, FSR, and current sensing	100% when using the two-stage imaging method with a second pass

It should be noted that the failures and inconsistencies observed in this work are mainly due to limitations in the imaging system's accuracy and the tolerances within the tooling assembly. The most prevalent failure mode in the automated screw disassembly is the screwdriver slipping away from the center of the cross-recessed screw when attempting to spin into the correct orientation after the force sensor detects contact with the part. This is related to inaccuracy with the deprojection system, as trying to seat the driver off-center would make the screwdriver re-orientation process more likely to slip. These limitations combined to prevent the testing and extraction of smaller-diameter cross-recessed fasteners. The stereoscopic camera's vision system introduced errors in its deprojection model, attributable to an inherent 2% depth sensing error specified by the manufacturer and the inability to image objects closer than approximately 280 mm. An empirical compensation method was applied to enhance target positioning accuracy. Testing was conducted to derive a 3D calibration curve, which compensated for the X and Y distances (relative to the camera imagers) as a function of deprojection distance. This curve was developed for the camera system's valid operational range (280–500 mm), where the DCNN model demonstrated reasonable target detection confidence. The calibration curve was constructed using three targets arranged vertically on a placard with known dimensions. The placard was evaluated at six positions across four depths relative to the RGB imager. Data were collected for 15 min at each position to capture a broad range of discrete detection coordinates.

The variability in the position of the tool center point defined on the electric screwdriver was estimated to be between 0.5 and 1.0 mm, which limited the overall process testing to the larger CRS targets discussed in this work. The inconsistency introduced as the tool changes orientation would necessitate a screwdriver with less variability in the chuck design, which secures the bit. This issue warrants further investigation for a more robust tooling design. Overall, the failures introduced by the prototype tooling and imaging system could be mitigated through a process design incorporating an iterative imaging technique, albeit at the expense of higher cycle times and lower first-pass yield.

5. Conclusions

This article describes a novel integration effort to combine an advanced DCNN vision system with robotic simulation software (RoboDK) that emulates a physical robot. The robot is equipped with a tool suite designed specifically for extracting cross-recessed screws. An innovative software package was developed that interfaces with all the physical components, including the tooling, camera system, UR5 robot, robot controller, and simulation software that communicates with the physical UR5. The custom-built modular software package can run the DCNN, interface with the RoboDK simulation software, and manage the physical tooling through serial communication.

The system was tested extensively, and the results are presented in the results and evaluation section from the camera's perspective, as well as the DCNN operation and functionality of the simulation software. The software suite, which bridges the state-based control of the robot to the control of all other pieces of equipment, was able to convert API calls received and issue driver commands to the robot while synchronously providing real-time feedback. Additionally, a substantial amount of work went into driving several evolutions of state-based logic to formulate an overall process for CRS extraction. Ultimately, it was proven that the system could consistently extract most 12.7 mm diameter cross-recessed screws from a test artifact with high consistency after a single cycle of extraction process execution. Using method #1, which incorporated only a single imaging stage for all targets without adjusting the camera position, a first-pass yield typically ranging between 44 and 67% was achieved. When utilizing method #2, which allowed the system to reimage the targets on the stepped artifact from multiple positions, a first-pass yield of 78–89% and a second-pass yield of 100% was achieved.

This work encountered several limitations and identified opportunities for more refined study, particularly regarding equipment, software implementation, and testing constraints. Using consumer off-the-shelf components, like the motor controller, screwdriver, and force-sensing resistor (FSR), while cost-effective and easy to implement, led to inconsistent readings. This inconsistency required the state-based logic to rely on trend observation rather than individual readings for real-time decision-making. The DCNN was trained on a limited dataset of approximately 1000 images and three sets of cross-recessed fasteners. This restricted the vision system's confidence and versatility, especially for larger targets and perpendicular viewing angles. Adding more pictures, especially ones taken at distances corresponding to the 300–500 mm detection range of the stereoscopic camera systems, would improve the detection accuracy of the DCNN model. Adding a higher-precision camera and more precise positioning methods would also enhance the detection accuracy. Analyzing the potential impact of different materials and surface treatments on detection is also essential to consider in future work. Given the project's scope and the extensive integration effort, several calibration methods, like ChArUco calibration for the stereoscopic camera, were only moderately explored. Implementing these could refine the TCP in the robotic simulation software, improving target deprojection and fastener localization in the robot's workspace.

Author Contributions: Conceptualization, M.K.J. and A.C.; methodology, M.K.J. and A.C.; software, A.C.; validation, A.C. and M.K.J.; formal analysis, A.C. and M.K.J.; investigation, A.C. and M.K.J.; resources, M.K.J.; data curation, A.C.; writing—original draft preparation, A.C. and M.K.J.; writing—review and editing, A.C. and M.K.J.; visualization, A.C.; supervision, M.K.J.; project administration, M.K.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in this article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wegener, K.; Andrew, S.; Raatz, A.; Droder, K.; Herrmann, C. Disassembly of electric vehicle batteries using the example of the Audi Q5 hybrid system. *Procedia CIRP* **2014**, *23*, 155–160. [CrossRef]
2. Duflou, J.R.; Seliger, G.; Kara, S.; Umeda, Y.; Ometto, A.; Willems, B. Efficiency and feasibility of product disassembly: A case-based study. *CIRP Ann.* **2008**, *57*, 583–600. [CrossRef]
3. Li, J.; Barwood, M.; Rahimifard, S. Robotic disassembly for increased recovery of strategically important materials from electrical vehicles. *Robot. Comput.-Integr. Manuf.* **2018**, *50*, 203–212. [CrossRef]
4. Wegener, K.; Chen, W.H.; Dietrich, F.; Droder, K.; Kara, S. Robot assisted disassembly for the recycling of electric vehicle batteries. *Procedia CIRP* **2015**, *29*, 716–721. [CrossRef]
5. Kristensen, C.B.; Sorensen, F.A.; Nielsen, H.B.; Andersen, M.S.; Bendtsen, S.P.; Bogh, S. Towards a robot simulation framework for e-waste disassembly using reinforcement learning. *Procedia Manuf.* **2019**, *38*, 225–332. [CrossRef]
6. Chen, W.H.; Foo, G.; Kara, S.; Pagnucco, M. Application of a multi-head tool for robotic disassembly. *Procedia CIRP* **2020**, *90*, 630–635. [CrossRef]
7. Vongbunyong, S.; Kara, S.; Pagnucco, M. Application of cognitive robotics in disassembly of products. *CIRP Ann.* **2013**, *62*, 31–34. [CrossRef]
8. Li, R.; Pham, D.T.; Huang, J.; Tan, Y.; Qu, M.; Wang, Y.; Kerin, M.; Jiang, K.; Su, S.; Ji, C.; et al. Unfastening of hexagonal headed screws by a collaborative robot. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1455–1468. [CrossRef]
9. DiFilippo, N.M.; Jouaneh, M.K. A system combining force and vision sensing for automated screw removal on laptops. *IEEE Trans. Autom. Sci. Eng.* **2017**, *15*, 887–895. [CrossRef]
10. Chen, J.; Liu, Z.; Xu, J.; Yang, C.; Chu, H.; Cheng, Q. A Novel Disassembly Strategy of Hexagonal Screws Based on Robot Vision and Robot-Tool Cooperated Motion. *Appl. Sci.* **2023**, *13*, 251. [CrossRef]

11. Yildiz, E.; Wörgötter, F. DCNN-based screw detection for automated disassembly processes. In Proceedings of the 2019 15th International Conference on Signal-Image Technology & Internet-Based Systems, Sorrento, Italy, 26–29 November 2019; IEEE: Sorrento, Italy, 2019; pp. 187–192. [CrossRef]
12. Brogan, D.P.; DiFilippo, N.M.; Jouaneh, M.K. Deep learning computer vision for robotic disassembly and servicing applications. *Array* **2021**, *12*, 100094. [CrossRef]
13. Karthikeyan, M.; Subashini, S.T. Automated object detection of mechanical fasteners using faster region based convolutional neural networks. *Int. J. Electr. Comput. Eng. IJECE* **2021**, *11*, 5430–5437. [CrossRef]
14. Foo, G.; Kara, S.; Pagnucco, M. Screw detection for disassembly of electronic waste using reasoning and re-training of a deep learning model. *Procedia CIRP* **2021**, *98*, 666–671. [CrossRef]
15. Mangold, S.; Steiner, C.; Friedmann, M.; Fleischer, J. Vision-based screw head detection for automated disassembly for remanufacturing. *Procedia CIRP* **2022**, *105*, 1–6. [CrossRef]
16. Zhang, X.; Eltouny, K.; Liang, X.; Behdad, S. Automatic Screw Detection and Tool Recommendation System for Robotic Disassembly. *J. Manuf. Sci. Eng.* **2023**, *145*, 031008. [CrossRef]
17. DiFilippo, N.M.; Jouaneh, M.K.; Jedson, A.D. Optimizing Automated Detection of Cross-Recessed Screws in Laptops Using a Neural Network. *Appl. Sci.* **2024**, *14*, 6301. [CrossRef]
18. Karbouj, B.; Topalian-Rivas, G.; Krüger, J. Comparative Performance Evaluation of One-Stage and Two-Stage Object Detectors for Screw Head Detection and Classification in Disassembly Processes. *Procedia CIRP* **2024**, *122*, 527–532. [CrossRef]
19. Zhu, C.; Zhang, J.; Xiao, J. Designing of an intelligent robotic system for disassembling screw connections of electric vehicle batteries based on the improved YOLOV5 algorithm. In Proceedings of the 2024 3rd International Symposium on Control Engineering and Robotics (ISCER'24), Changsha, China, 24–26 May 2024; Association for Computing Machinery: New York, NY, USA, 2024; pp. 1–6.
20. Peng, Y.; Li, W.; Liang, Y.; Pham, D. Robotic disassembly of screws for end-of-life product remanufacturing enabled by deep reinforcement learning. *J. Clean. Prod.* **2024**, *439*, 140863. [CrossRef]
21. Zhang, Y.; Zhang, H.; Wang, Z.; Zhang, S.; Li, H.; Chen, M. Development of an Autonomous, Explainable, Robust Robotic System for Electric Vehicle Battery Disassembly. In Proceedings of the 2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Seattle, WA, USA, 28–30 June 2023; pp. 409–414. [CrossRef]
22. Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. *Robotics Modeling, Planning, and Control*; Springer: Berlin/Heidelberg, Germany, 2009.
23. Clark, A. Automated Extraction of Fasteners Using Digital Twin Robotics Software and DCNN Vision System. Master's Thesis, University of Rhode Island, Kingston, RI, USA, 2022. [CrossRef]